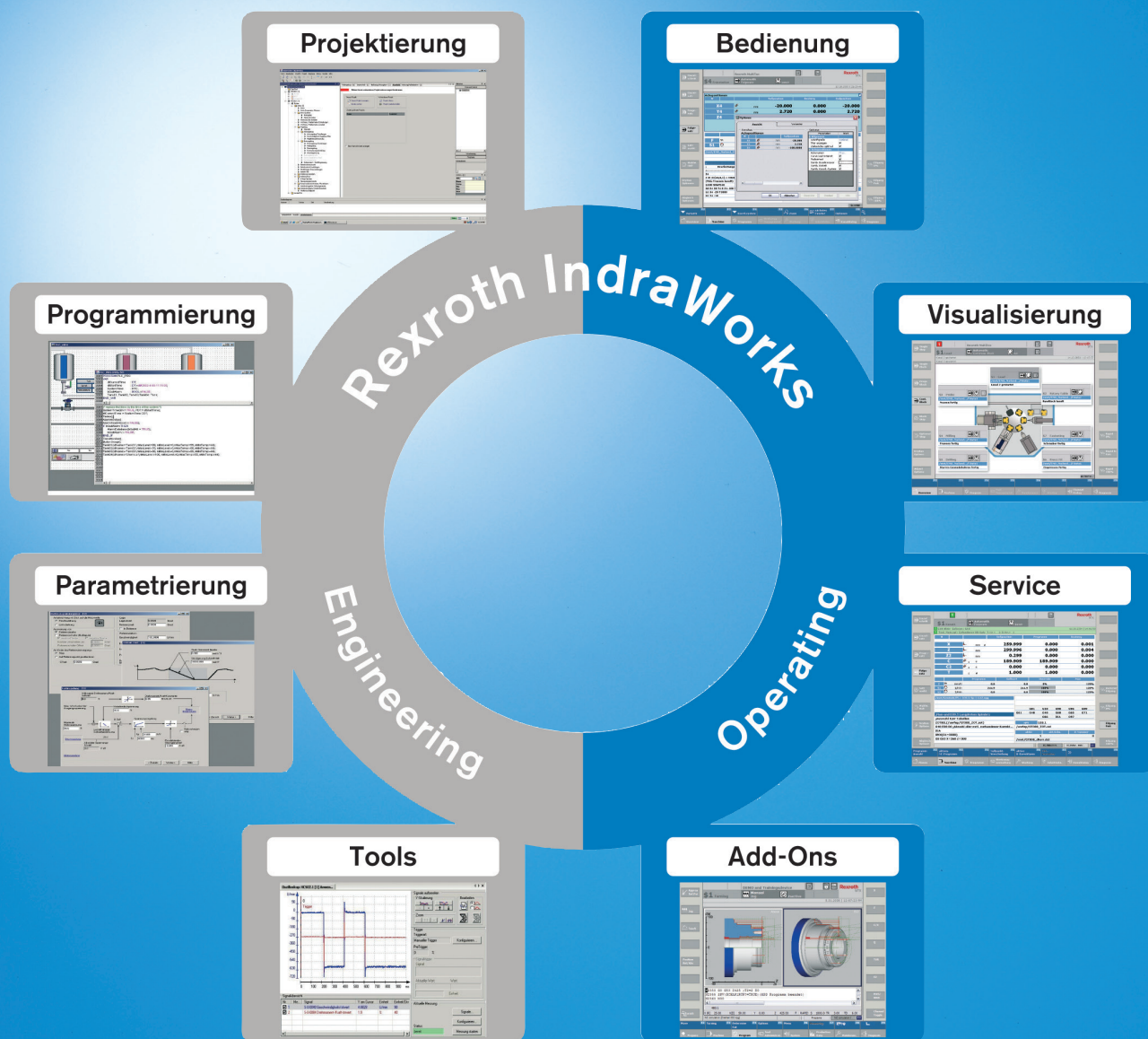


Rexroth IndraWorks 13VRS WinStudio 7.3

R911336882
Edition 01

Reference Book



Title	Rexroth IndraWorks 13VRS WinStudio 7.3								
Type of Documentation	Reference Book								
Document Typecode	DOK-IWORKS-WINSTUD*V13-AP01-EN-P								
Internal File Reference	RS-745c358d9b032dba0a6846a501cacaba-1-en-US-5								
Purpose of Documentation	This "User Manual and Technical Reference Book" has been designed to achieve the best results using the "Rexroth WinStudio"™ software. The document provides technical information and step-by-step instructions for the creation of web-capable HMI/SCADA programs.								
Record of Revision	<table><tr><th>Edition</th><th>Release Date</th><th>Notes</th></tr><tr><td>Edition 01</td><td>12.2013</td><td>First edition 13 VRS</td></tr></table>			Edition	Release Date	Notes	Edition 01	12.2013	First edition 13 VRS
Edition	Release Date	Notes							
Edition 01	12.2013	First edition 13 VRS							
Copyright	© Bosch Rexroth AG 2013 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/ DC-IA/EAY2 TBe (vha/PiGe)								
Note	This document has been printed on chlorine-free bleached paper.								

Table of Contents

	Page
1 About this Documentation.....	19
1.1 Validity of this Documentation.....	19
1.2 Documentation Structure.....	19
1.3 Using the Safety Instructions.....	20
1.3.1 Safety Instructions – Structure.....	20
1.3.2 Explaining Signal Words and Safety Alert Symbol.....	21
1.3.3 Symbols Used.....	21
1.4 Conventions Used in this Documentation.....	21
1.5 Required and Supplementing Documentation.....	22
2 About this Application.....	25
2.1 About this Application, General Information.....	25
2.2 Product Overview.....	25
2.3 Product Features.....	26
2.4 How the Software Works.....	27
2.4.1 Internal Structure and Data Flow.....	27
2.4.2 Executing/Switching Modules.....	30
2.4.3 Executing/Switching the Background Task.....	34
3 Installation.....	37
3.1 Installation, General Information.....	37
3.2 System Requirements.....	37
3.3 Installing the Software.....	38
3.4 Starting the Engineering Software.....	42
4 Licensing.....	43
4.1 License Protection Types.....	43
4.2 License Settings.....	43
4.3 License Types.....	44
4.4 Product Versions.....	45
4.5 WinStudio License Activation.....	46
4.6 Invalid Licenses.....	47
5 Development Environment.....	49
5.1 The Development Environment, General Information.....	49
5.2 WinStudio Toolbars.....	49
5.2.1 WinStudio Toolbars, Customizing.....	49
5.2.2 WinStudio Execution Functions Toolbar.....	50
5.2.3 WinStudio Alignment Functions Toolbar.....	51
5.2.4 WinStudio Tag Properties Toolbar.....	52
5.2.5 WinStudio Object Properties Toolbar.....	52
5.2.6 WinStudio Table Functions Toolbar.....	53
5.2.7 WinStudio Debug Functions Toolbar.....	53

Table of Contents

	Page
5.2.8 WinStudio Dynamic Functions Toolbar.....	54
5.3 Project Explorer.....	55
5.3.1 Project Explorer, General Information.....	55
5.3.2 "WinStudio" Folder.....	55
5.3.3 "Variables" Folder.....	56
5.3.4 "Tasks" Folder.....	57
5.3.5 Communication Folder.....	58
5.4 Configuring and Testing an Application.....	59
5.4.1 Application Verification.....	59
5.4.2 Starting the Application.....	60
5.4.3 Stopping the Application.....	60
5.4.4 WinStudio Target Platform.....	61
5.4.5 Project Settings.....	62
Information Tab.....	62
Options Tab.....	62
Options Tab, General Information.....	62
Saving the Alarm History to an External Database.....	63
Configuring a Default Database for All Task Histories.....	64
Database Configuration.....	65
Viewer Tab.....	69
Viewer Tab, General Information.....	69
Keyboard.....	73
Virtual Keyboard.....	73
Communication Tab.....	74
Preferences Tab.....	76
Web Tab.....	78
5.5 Tools.....	82
5.6 Screen/Worksheet Editor.....	84
5.7 Database Spy.....	84
5.8 LogWin Output Window.....	85
5.9 Cross Reference List.....	87
5.10 Status Bar.....	87
6 WinStudio Projects.....	89
6.1 Creating a New Project.....	89
6.1.1 Creating a New Project, General Information.....	89
6.1.2 Target Platforms and Product Types.....	90
6.1.3 Changing the Target System of an Existing Project.....	91
6.2 E-Mail Configuration.....	92
6.3 FTP Configuration.....	94
6.4 Project Execution under Windows Services.....	95
7 Working with Tags.....	105
7.1 Tags and the Project Database.....	105
7.1.1 Tags and the Project Database, General Information.....	105

Table of Contents

	Page
7.1.2 Application Variables.....	106
Application Variables, General Information.....	106
Extending the Application Variables Datasheet.....	107
7.1.3 Classes Folder.....	108
7.1.4 Distributed Variables.....	109
7.1.5 Internal Variables.....	110
7.2 Creating Tags.....	111
7.2.1 Tag Name Syntax.....	111
7.2.2 Choosing the Tag Type.....	111
7.2.3 Choosing the Tag Data Type.....	113
7.2.4 Using Array Tags.....	114
7.2.5 Using Classes.....	117
7.2.6 Using Indirect Tags (Pointers).....	119
7.2.7 Choosing the Tag Scope.....	120
7.3 Creating Database Tags.....	121
7.3.1 Creating Tags on the Datasheet.....	121
7.3.2 Creating Tags "On-the-Fly".....	122
7.3.3 Editing Tags.....	122
7.4 Creating Classes.....	123
7.5 Setting Tag Properties.....	124
7.5.1 Tag Properties and Parameters.....	124
7.5.2 Using Tag Properties: Alarms.....	126
7.5.3 Using Tag Properties: History.....	129
7.5.4 List of Tag Properties.....	129
7.6 Using Tags in the Project.....	134
7.7 Deleting a Tag from the Project Database.....	135
7.8 Renaming and Removing Tags.....	135
7.8.1 Replace Tags Globally	135
7.8.2 Replacing Tags in a Document or Screen Object.....	136
7.8.3 Removing Unused Tags from the Project Database.....	137
7.8.4 Tags Database Reset.....	138
7.8.5 Using Available Tags	139
7.8.6 Changing Tag Properties.....	140
7.9 Importing an External Database.....	140
7.9.1 Import Wizard.....	140
7.9.2 Importing from.....	143
Importing from Other Project Databases.....	143
Importing from OPC Server Databases.....	145
Importing from CSV Databases.....	146
Importing from ODBC Databases.....	147
7.10 Tag Integration.....	148
7.10.1 Tag Integration, General Information.....	148
7.10.2 Sharing Tags on IndraLogic Devices.....	149
8 Screens and Graphics.....	155
8.1 Screens and Graphics, General Information.....	155

Table of Contents

	Page
8.2	WinStudio Folders..... 156
8.2.1	WinStudio Folder..... 156
	WinStudio Folder, General Information..... 156
	Screen Attributes Dialog..... 156
	Background Color or Picture of a WinStudio Screen..... 160
8.2.2	Screen Groups Folder..... 161
8.2.3	Web Pages Folder..... 162
8.3	Using Screen Objects and Animations..... 164
8.3.1	Editing Screen Objects..... 164
	Editing Screen Objects, General Information..... 164
	Editing Object Properties..... 164
	Replacing Project Tags in a Document or Screen Object..... 164
	Arranging Objects..... 166
	Locking Objects in the Screen Editor..... 166
	Undo..... 166
	Grid Settings..... 167
8.3.2	Static Objects..... 167
	Static Objects, General Information..... 167
	Line Object..... 167
	Open Polygon Object..... 168
	Closed Polygon Object..... 169
	Rectangle Object..... 170
	Rounded Rectangle Object..... 171
	Ellipse Object..... 172
	Adding Graphics to a WinStudio Screen..... 173
8.3.3	Active Objects..... 174
	Active Objects, General Information..... 174
	Text Object..... 175
	Button Object..... 176
	Pushbutton Object..... 178
	Checkbox Object..... 181
	Radio Button Object..... 184
	Operating Modes for Checkbox and Radio Button Objects..... 186
	Combo Box Object..... 188
	List Box Object..... 192
	Smart Message Object..... 195
8.3.4	WinStudio Libraries..... 199
	Symbols..... 199
	Symbols - General Information..... 199
	Saving Own Project Symbols..... 200
	ActiveX Control..... 208
	.NET Controls..... 214
	Adding a Link to an External Graphic File..... 219
8.3.5	Applying Animations to Screen Objects..... 220
	Applying Animations to Screen Objects, General Information..... 220
	Command Animation..... 220

Table of Contents

	Page
Hyperlink Animation.....	224
Bargraph Animation.....	224
Text Data Link Animation.....	225
Color Animation.....	227
Position Animation.....	229
Resize Animation.....	231
Rotation Animation.....	232
8.4 Formatting Screen Objects.....	233
8.4.1 Formatting Screen Objects, General Information.....	233
8.4.2 "Move to Foreground" and "Move to Background".....	234
8.4.3 "To the Foreground" and "To the Background", by One Level	235
8.4.4 "Group " and "Ungroup Objects".....	235
8.4.5 "Align", "Center" and "Distribute".....	236
8.4.6 "Rotating" and "Mirroring".....	238
"Rotating".....	238
"Mirroring Vertically".....	238
"Mirroring Horizontally".....	238
8.4.7 Adjusting the Position.....	239
8.4.8 "Same Width" / "Same Height" for Selected Objects.....	239
8.4.9 Fill Color.....	240
8.4.10 Line Color.....	241
8.4.11 Color Interface.....	241
8.4.12 Fonts.....	244
9 Alarms, Events and Trends.....	247
9.1 Alarms, Events and Trends, General Information.....	247
9.2 Alarm Worksheet.....	247
9.2.1 Alarm Worksheet, General Information.....	247
9.2.2 Alarm Worksheet Header.....	248
Alarm Worksheet Header, General Information.....	248
Email Settings for Alarm Worksheet.....	250
Advanced Settings for Alarm Worksheet.....	252
9.2.3 Alarm Worksheet Body.....	253
9.2.4 Saving the Alarm History to an External Database.....	254
9.2.5 Format of the Alarm History.....	255
9.3 Events and Event Logs.....	258
9.3.1 Events and Event Logs, General Information.....	258
9.3.2 Enabling the Event Logger.....	258
9.3.3 Saving the Event Log to an External Database.....	259
9.3.4 Format of the Event Log.....	260
9.4 Alarm/Event Control Object.....	262
9.5 Trend Worksheet.....	269
9.5.1 Trend Worksheet, General Information.....	269
9.5.2 Converting Trend History Files from Binary to Text.....	272
9.5.3 Converting Trend History Files from Text to Binary.....	273
9.5.4 Creating Batch History.....	273

Table of Contents

	Page
9.5.5	Setting the Trend Database..... 276
9.6	Trend Display..... 277
9.6.1	Trend Display, General Information..... 277
9.6.2	Trend Display Operating Elements..... 278
9.6.3	Trend Object Properties..... 281
	Trend Object Properties, General Information..... 281
	Points..... 282
	Points Dialog, General Information..... 282
	Pen Style..... 283
	Options..... 285
	SPC..... 286
	Axes..... 287
	Toolbar..... 291
	Data Source..... 292
	Legend..... 294
	Advanced..... 295
9.6.4	Using a Text File as Data Source..... 298
9.6.5	Using a Database as Data Source..... 301
9.7	Table Object..... 305
9.7.1	Table Object, General Information..... 305
9.7.2	Columns Dialog..... 307
9.7.3	Data Dialog..... 309
9.7.4	Advanced Dialog..... 311
10	Background Tasks..... 315
10.1	Background Tasks, General Information..... 315
10.2	Alarm Worksheet, General Information..... 315
10.3	Trend Worksheet, General Information..... 316
10.4	Recipe Worksheet..... 319
10.5	Report Worksheet..... 320
10.6	ODBC Worksheet..... 321
10.7	Math Worksheet..... 324
10.8	Script Worksheet..... 324
10.8.1	Script Worksheet, General Information..... 324
10.8.2	Script Worksheet Startup..... 325
10.9	Scheduler Worksheet..... 326
10.10	Database/ERP Worksheet..... 328
11	Communication with Other Devices..... 333
11.1	Communication with Other Devices, General Information..... 333
11.2	Configuring Direct Communication with a Remote Device..... 333
11.2.1	Configuring Direct Communication with a Remote Device, General Information..... 333
11.2.2	Driver Sheets..... 339
11.3	Configuring an OPC Client Connection to an OPC Server..... 342
11.4	Configuring an OPC UA Client Connection to an OPC UA Server..... 345

Table of Contents

	Page
11.5	Configuring a TCP/IP Client Connection to Another Project..... 351
11.6	Configuring a DDE Client Connection to a DDE Server..... 352
12	Project Security..... 357
12.1	Project Security, General Information..... 357
12.2	Security Modes..... 357
12.3	Security Access Levels..... 358
12.4	Using the Security System Configuration Wizard..... 359
12.5	Configuring Server Settings for Security Modes..... 362
12.5.1	Configuring Server Settings for Security Modes, General Information..... 362
12.5.2	Extending the LDAP Schema to Allow Saving of Security Rights..... 365
12.6	Creating and Configuring Groups..... 370
12.7	Creating and Configuring Users..... 374
12.8	Security System Dialog..... 375
12.9	Backing up the Security System Configuration..... 377
12.10	Logging On/Off..... 378
12.11	Blocking or Unblocking Users..... 379
12.12	Password Protection for Screens, Symbols and Worksheets..... 379
13	Translation of User Texts..... 381
13.1	Translation of User Texts, General Information..... 381
13.2	Adding a Project Language..... 381
13.3	Translating Application Texts to Additional Languages..... 382
13.4	Activating a Language at Runtime..... 382
13.4.1	Activating a Language at Runtime, General Information..... 382
13.4.2	Setting the Project Language at Program Start..... 383
13.4.3	Setting the Project Language at runtime..... 383
13.5	Disabling Translation of Selected Screen Objects..... 384
14	Testing and Debugging Projects..... 385
14.1	Output (LogWin)..... 385
14.2	Database Spy..... 387
14.3	Using the LogWin Module..... 387
14.4	Using Remote Tools..... 388
14.4.1	Remote Database Spy..... 388
14.4.2	Remote LogWin..... 389
15	Provision as a Web Application..... 391
15.1	Introduction to Thin Clients..... 391
15.2	Creating a Simple Thin Client Program..... 393
15.3	Underlying Technology..... 396
15.4	ISSymbol Control Layer..... 397
15.5	Examples of Client/Server Architecture..... 399
15.6	Configuring the Data Server..... 403
15.7	Configuring a Web Server as Host for Project Pages..... 407

Table of Contents

	Page
15.8	Installing the Web Tunneling Gateway..... 410
15.9	Configuring the Thin Client..... 413
15.10	Implementing Security for Web-based Applications..... 417
15.11	Port Usage..... 421
15.12	Exercise: Viewing the Project on the Web..... 422
16	Downloading to a Remote Device..... 425
16.1	Configuring the Target System..... 425
16.2	Configuring the Development Station..... 426
16.3	Automatically Running a Project..... 428
17	Database Interface..... 429
17.1	Database Interface, General Information..... 429
17.2	SQL Relational Databases..... 430
17.3	Linking the Database Through a Remote DB Provider..... 431
17.4	Studio Database Gateway..... 432
17.5	Database Configuration..... 438
17.6	Configuring a Default Database for All Task Histories..... 442
17.7	Database Debugging..... 443
17.8	Appendices..... 447
17.8.1	Using ODBC Databases..... 447
17.8.2	Using Microsoft SQL Server..... 448
17.8.3	Using ORACLE Databases..... 449
17.8.4	Using Microsoft Access Databases..... 451
17.8.5	Using SQL Server CE..... 453
	Using SQL Server CE, General Information..... 453
	Installing SQL Server on a Windows Embedded Device..... 454
17.8.6	Using Sybase..... 455
17.8.7	Using Microsoft Excel..... 456
17.8.8	Using MySQL..... 458
18	Troubleshooting and FAQs..... 461
18.1	General Troubleshooting..... 461
18.2	Frequently Asked Questions..... 462
19	Appendix: WinStudio-Internal Scripting Language..... 471
19.1	Appendix: Internal Scripting Language, General Information..... 471
19.2	List of Available Functions..... 471
19.3	Structure of the Functional Descriptions..... 482
19.4	Log Message Functions..... 483
19.4.1	Log Message Functions – General Information..... 483
19.4.2	Trace..... 483
19.5	Arithmetic Functions..... 484
19.5.1	Arithmetic Functions – General Information..... 484
19.5.2	Arithmetic Operators..... 484

Table of Contents

	Page
19.5.3 Abs.....	485
19.5.4 Div.....	485
19.5.5 Format.....	486
19.5.6 GetBit.....	489
19.5.7 Mod.....	490
19.5.8 Pow.....	490
19.5.9 ResetBit.....	491
19.5.10 Round.....	492
19.5.11 SetBit.....	492
19.5.12 Sqrt.....	493
19.5.13 Swap16.....	493
19.5.14 Swap32.....	494
19.5.15 Trunc.....	494
19.6 Statistical Functions.....	495
19.6.1 Statistical Functions – General Information.....	495
19.6.2 Avg.....	495
19.6.3 Max.....	496
19.6.4 Min.....	497
19.6.5 Rand.....	498
19.7 Logarithmic Functions.....	498
19.7.1 Logarithmic Functions – General Information.....	498
19.7.2 Exp.....	498
19.7.3 Log.....	499
19.7.4 Log10.....	499
19.8 Logical functions.....	500
19.8.1 Logical functions – General Information.....	500
19.8.2 Logical Operators.....	500
19.8.3 False.....	501
19.8.4 If.....	501
19.8.5 Toggle.....	502
19.8.6 True.....	503
19.9 String Functions.....	503
19.9.1 String Functions – General Information.....	503
19.9.2 Asc2Str.....	504
19.9.3 CharToValue.....	505
19.9.4 CharToValueW.....	505
19.9.5 ClassMembersToStrVector.....	506
19.9.6 NCopy.....	507
19.9.7 Num.....	508
19.9.8 Str.....	508
19.9.9 Str2Asc.....	509
19.9.10 StrCompare.....	509
19.9.11 StrCompareNoCase.....	510
19.9.12 StrFromInt.....	510
19.9.13 StrFromReal.....	511
19.9.14 StrFromTime.....	512

Table of Contents

	Page
19.9.15 StrGetElement.....	513
19.9.16 StrLeft.....	513
19.9.17 StrLen.....	514
19.9.18 StrLower.....	514
19.9.19 StrRChr.....	514
19.9.20 StrRight.....	515
19.9.21 StrSetElement.....	515
19.9.22 StrStr.....	516
19.9.23 StrStrPos.....	516
19.9.24 StrTrim.....	517
19.9.25 StrTrimAll.....	518
19.9.26 StrUpper.....	518
19.9.27 ValueToChar.....	519
19.9.28 ValueWToChar.....	519
19.10 Date and Time Functions.....	520
19.10.1 Date & Time functions – General Information.....	520
19.10.2 ClockGetDate.....	520
19.10.3 ClockGetDayOfWeek.....	521
19.10.4 ClockGetTime.....	522
19.10.5 DateTime2Clock.....	522
19.10.6 GetClock.....	523
19.10.7 Hour2Clock.....	523
19.10.8 SetSystemDate.....	524
19.10.9 SetSystemTime.....	524
19.11 Trigonometric Functions.....	524
19.11.1 Trigonometric Functions – General Information.....	524
19.11.2 ACos.....	525
19.11.3 ASin.....	525
19.11.4 ATan.....	526
19.11.5 Cos.....	526
19.11.6 Cot.....	526
19.11.7 Pi.....	527
19.11.8 Sin.....	527
19.11.9 Tan.....	527
19.12 Screen Functions.....	528
19.12.1 Screen Functions – General Information.....	528
19.12.2 Close.....	528
19.12.3 Open.....	529
19.12.4 OpenPrevious.....	531
19.12.5 ShowInplaceInput.....	532
19.12.6 ShowMessageBox.....	533
19.13 Security Functions.....	534
19.13.1 Security Functions – General Information.....	534
19.13.2 BlockUser.....	535
19.13.3 CheckESign.....	535
19.13.4 CreateUser.....	536

Table of Contents

	Page
19.13.5 ExportSecuritySystem.....	537
19.13.6 GetSecuritySystemStatus.....	538
19.13.7 GetUserFullName.....	539
19.13.8 GetUserNames.....	540
19.13.9 GetUserPwdAging.....	541
19.13.10 GetUserState.....	541
19.13.11 ImportSecuritySystem.....	542
19.13.12 RemoveUser.....	543
19.13.13 SetPassword.....	544
19.13.14 UnblockUser.....	545
19.14 Module Activity Functions.....	545
19.14.1 Module Activity Functions – General Information.....	545
19.14.2 AppActivate.....	547
19.14.3 AppIsRunning.....	548
19.14.4 AppPostMessage.....	549
19.14.5 AppSendKeys.....	550
19.14.6 CleanReadQueue.....	550
19.14.7 CloseSplashWindow.....	550
19.14.8 DisableMath.....	551
19.14.9 EnableMath.....	551
19.14.10 EndTask.....	551
19.14.11 ExitWindows.....	552
19.14.12 IsScreenOpen.....	553
19.14.13 IsTaskRunning.....	553
19.14.14 IsViewerInFocus.....	554
19.14.15 KeyPad.....	554
19.14.16 LogOff.....	556
19.14.17 LogOn.....	556
19.14.18 Math.....	557
19.14.19 PostKey.....	557
19.14.20 Recipe.....	558
19.14.21 Report.....	558
19.14.22 RunGlobalProcedureAsync.....	560
19.14.23 RunGlobalProcedureAsyncGetStatus.....	561
19.14.24 RunGlobalProcedureOnFalse.....	563
19.14.25 RunGlobalProcedureOnServer.....	564
19.14.26 RunGlobalProcedureOnTrigger.....	565
19.14.27 RunGlobalProcedureOnTrue.....	565
19.14.28 RunVBScript.....	566
19.14.29 SecureViewerReload.....	567
19.14.30 SendKeyObject.....	568
19.14.31 SetAppPath.....	570
19.14.32 SetViewerInFocus.....	570
19.14.33 SetViewerPos.....	571
19.14.34 ShutDown.....	572
19.14.35 StartTask.....	572

Table of Contents

	Page
19.14.36 ViewerPostMessage.....	573
19.14.37 Wait.....	574
19.14.38 WinExec.....	574
19.14.39 WinExecIsRunning.....	576
19.15 File functions.....	577
19.15.1 File functions - General Information.....	577
19.15.2 DeleteOlderFiles.....	577
19.15.3 DirCreate.....	578
19.15.4 DirDelete.....	579
19.15.5 DirLength.....	579
19.15.6 DirRename.....	580
19.15.7 FileCopy.....	581
19.15.8 FileDelete.....	581
19.15.9 FileLength.....	582
19.15.10 FileRename.....	582
19.15.11 FileWrite.....	583
19.15.12 FindFile.....	584
19.15.13 FindPath.....	585
19.15.14 GetFileAttributes.....	585
19.15.15 GetFileTime.....	586
19.15.16 GetHstInfo.....	586
19.15.17 GetLine.....	587
19.15.18 HST2TXT.....	590
19.15.19 HST2TXTIsRunning.....	591
19.15.20 LookupContains.....	592
19.15.21 LookupGet.....	593
19.15.22 LookupLoad.....	593
19.15.23 PDFCreate.....	594
19.15.24 Print.....	595
19.15.25 RDFileN.....	595
19.15.26 WebGetFile.....	596
19.16 Graphic Functions.....	597
19.16.1 Graphic functions – General Information.....	597
19.16.2 AutoFormat.....	597
19.16.3 GetScrInfo.....	598
19.16.4 PrintSetup.....	599
19.16.5 PrintWindow.....	599
19.16.6 ResetDecimalPointsTable.....	601
19.16.7 RGBColor.....	601
19.16.8 RGBComponent.....	602
19.16.9 SaveScreenShot.....	602
19.16.10 SetDecimalPoints.....	604
19.16.11 SetDisplayUnit.....	605
19.16.12 SetTagDisplayUnit.....	606
19.17 Translation Functions.....	606
19.17.1 Translation Functions – General Information.....	606

Table of Contents

	Page
19.17.2 Ext.....	607
19.17.3 SetLanguage.....	607
19.17.4 SetTranslationFile.....	608
19.18 Multimedia Functions.....	608
19.18.1 Multimedia Functions – General Information.....	608
19.18.2 Play.....	609
19.19 System Information Functions.....	609
19.19.1 System Information Functions – General Information.....	609
19.19.2 DbVersion.....	611
19.19.3 GetAppHorizontalResolution.....	611
19.19.4 GetAppPath.....	612
19.19.5 GetAppVerticalResolution.....	612
19.19.6 GetComputerIP.....	612
19.19.7 GetComputerName.....	613
19.19.8 GetCursorX.....	613
19.19.9 GetCursorY.....	613
19.19.10 GetDisplayHorizontalResolution.....	614
19.19.11 GetDisplayVerticalResolution.....	614
19.19.12 GetFreeMemoryCE.....	614
19.19.13 GetHardKeyModel.....	615
19.19.14 GetHardKeySN.....	615
19.19.15 GetIPAll.....	616
19.19.16 GetMemoryCE.....	616
19.19.17 GetNetMACID.....	617
19.19.18 GetOS.....	618
19.19.19 GetPrivateProfileString.....	618
19.19.20 GetProductPath.....	619
19.19.21 GetRegValue.....	619
19.19.22 GetRegValueType.....	620
19.19.23 GetServerHostName.....	621
19.19.24 GetTickCount.....	621
19.19.25 InfoAppAlrDir.....	622
19.19.26 InfoAppHSTDir.....	622
19.19.27 InfoDiskFree.....	622
19.19.28 InfoResources.....	623
19.19.29 IsActiveXReg.....	623
19.19.30 IsAppChangedOnServer.....	624
19.19.31 NoInputTime.....	625
19.19.32 ProductVersion.....	625
19.19.33 RegSaveCE.....	626
19.19.34 ReloadAppFromServer.....	626
19.19.35 SaveAlarmFile.....	627
19.19.36 SetAppAlarmPath.....	628
19.19.37 SetAppHSTPath.....	628
19.19.38 SetDateFormat.....	629
19.19.39 SetKeyboardLanguage.....	629

Table of Contents

	Page
19.19.40 SetRegValue.....	630
19.19.41 SetWebConfig.....	631
19.19.42 SNMPGet.....	633
19.19.43 SNMPSet.....	634
19.19.44 WritePrivateProfileString.....	636
19.20 Tags Database Functions.....	636
19.20.1 Tags Database Functions – General Information.....	636
19.20.2 ExecuteAlarmAck.....	637
19.20.3 ForceTagChange.....	638
19.20.4 GetTagValue.....	638
19.20.5 SetTagValue.....	639
19.21 Loop Functions.....	640
19.21.1 Loop Functions – General Information.....	640
19.21.2 For ... Next.....	640
19.22 ODBC Functions.....	640
19.22.1 ODBC Functions – General Information.....	640
19.22.2 ODBCBeginTrans.....	642
19.22.3 ODBCBindCol.....	642
19.22.4 ODBCCanAppend.....	644
19.22.5 ODBCCanTransact.....	644
19.22.6 ODBCCanUpdate.....	644
19.22.7 ODBCClose.....	645
19.22.8 ODBCCommitTrans.....	645
19.22.9 ODBCDelete.....	646
19.22.10 ODBCExecuteSQL.....	646
19.22.11 ODBCInsert.....	647
19.22.12 ODBCIsBOF.....	648
19.22.13 ODBCIsDeleted.....	648
19.22.14 ODBCIsEOF.....	649
19.22.15 ODBCIsFieldNULL.....	649
19.22.16 ODBCIsFieldNullable.....	650
19.22.17 ODBCMove.....	650
19.22.18 ODBCMoveFirst.....	651
19.22.19 ODBCMoveLast.....	651
19.22.20 ODBCMoveNext.....	652
19.22.21 ODBCMovePrev.....	652
19.22.22 ODBCOpen.....	653
19.22.23 ODBCQuery.....	654
19.22.24 OBCRollBack.....	654
19.22.25 ODBCSetFieldNULL.....	655
19.22.26 ODBCSetFilter.....	656
19.22.27 ODBCSetSort.....	656
19.22.28 ODBCUnbindCol.....	657
19.22.29 ODBCUpdate.....	657
19.23 Email Functions.....	658
19.23.1 Email Functions – General Information.....	658

Table of Contents

	Page
19.23.2 CnfEmail.....	658
19.23.3 GetStatusSendEmailExt.....	661
19.23.4 SendEmail.....	661
19.23.5 SendEmailExt.....	662
19.24 Dial-up Functions.....	664
19.24.1 Dial-up Functions – General Information.....	664
19.24.2 DialError.....	664
19.24.3 DialGetClientIP.....	670
19.24.4 DialGetServerIP.....	672
19.24.5 DialStatus.....	673
19.24.6 DialUp.....	675
19.24.7 DialUpToCE.....	676
19.24.8 FindAllDevices.....	678
19.24.9 FindModem.....	678
19.24.10 HangUp.....	679
19.24.11 PhoneDialUp.....	680
19.24.12 PhoneDisableListen.....	680
19.24.13 PhoneEnableListen.....	681
19.24.14 PhoneHangUp.....	682
19.24.15 PhoneStatus.....	682
19.25 ActiveX and .NET control functions.....	683
19.25.1 ActiveX and .NET Control Functions – General Information.....	683
19.25.2 XGet.....	684
19.25.3 XRun.....	684
19.25.4 XSet.....	685
19.26 Event Logger Functions.....	686
19.26.1 Event Logger Functions – General Information.....	686
19.26.2 SendEvent.....	686
19.27 FTP Functions.....	687
19.27.1 FTP Functions – General Information.....	687
19.27.2 CnfFTP.....	687
19.27.3 FTPGet.....	688
19.27.4 FTPPut.....	690
19.27.5 FTPStatus.....	691
19.28 Database/ERP Functions.....	692
19.28.1 Database/ERP Functions – General Information.....	692
19.28.2 DBCursorClose.....	693
19.28.3 DBCursorColumnCount.....	694
19.28.4 DBCursorColumnInfo.....	694
19.28.5 DBCursorCurrentRow.....	695
19.28.6 DBCursorGetValue.....	696
19.28.7 DBCursorMoveTo.....	697
19.28.8 DBCursorNext.....	698
19.28.9 DBCursorOpen.....	699
19.28.10 DBCursorOpenSQL.....	700
19.28.11 DBCursorPrevious.....	702

Table of Contents

	Page
19.28.12 DBCursorRowCount.....	703
19.28.13 DBDelete.....	703
19.28.14 DBExecute.....	704
19.28.15 DBInsert.....	706
19.28.16 DBSelect.....	707
19.28.17 DBUpdate.....	709
19.28.18 SyncAlarm.....	710
19.28.19 SyncAlarmStatus.....	711
19.28.20 SyncEvent.....	711
19.28.21 SyncEventStatus.....	712
19.28.22 SyncTrend.....	713
19.28.23 SyncTrendStatus.....	714
20 Appendix: VBScript Language.....	715
20.1 Overview of VBScript.....	715
20.2 VBScript Interfaces in the Software.....	715
20.2.1 VBScript Interfaces in the Software - General Information.....	715
20.2.2 Global Procedures.....	716
20.2.3 Graphic Module.....	718
Graphic Script.....	718
Screen Script.....	720
Command Animation.....	721
ActiveX Events.....	722
20.2.4 Background Task.....	725
20.3 Language Reference.....	725
20.3.1 Language Reference – Overview.....	725
20.3.2 Operators.....	725
20.3.3 Constants.....	726
20.3.4 Objects and Collections.....	730
20.3.5 Properties.....	730
20.3.6 Statements.....	731
20.3.7 Methods.....	733
20.3.8 Functions.....	734
20.3.9 Keywords.....	735
20.3.10 Errors.....	735
20.3.11 List of System Tags.....	738
20.4 Tips & Tricks.....	740
20.4.1 Tips & Tricks – Overview.....	740
20.4.2 VBScript Editor IntelliSense.....	740
20.4.3 VBScript compared to VBA.....	742
20.4.4 Screen Events.....	744
20.4.5 MsgBox and InputBox Functions.....	744
20.4.6 VBScript Procedures.....	744
20.4.7 Creating Constants.....	746
20.4.8 Declaring Variables.....	746
20.4.9 Scope and Lifetime of Variables.....	747

Table of Contents

	Page
20.4.10 Boolean Tags and Boolean Variables.....	747
20.4.11 Writing Real Values to Integer Tags.....	748
20.4.12 Precedence of VBScript Operators.....	749
20.4.13 Logical Operator NOT.....	749
20.4.14 Using Conditional Statements.....	750
20.4.15 Looping with Code.....	752
20.4.16 Support for ActiveX Control Elements.....	754
20.4.17 Windows Embedded Support.....	755
21 Appendix: Driver Error Codes.....	757
22 Service and Support.....	759
Index.....	761

1 About this Documentation

1.1 Validity of this Documentation

This "User Manual and Technical Reference" has been designed to achieve the best results using the "Rexroth WinStudio"™ software. This document provides technical information and step-by-step instructions for the creation of web-capable HMI/SCADA programs.

Target group This "User Manual and Technical Reference" is a comprehensive document designed to provide useful information for both new and experienced users of "WinStudio".

- **New users:** This publication uses a step-by-step, hands-on approach to the project development process. Read the introduction describing the product's features and development environment.
- **Experienced users:** The document contains instructions for specialists as well as tips and information for troubleshooting.

Necessary qualification: Persons who can assess the work to be undertaken and recognize potential hazards due to their professional training, knowledge and experience as well as their understanding of relevant standards and regulations.

1.2 Documentation Structure

The information in this document is organized into the following sections:

- [chapter 2 "About this Application" on page 25](#): Includes a product overview, a list of important product features as well as information on the software functionality
- [chapter 3 "Installation" on page 37](#): Includes installation instructions for "Rexroth WinStudio", Engineering, EmbeddedView and CEView
- [chapter 4 "Licensing" on page 43](#): Describes the type of license protection, license settings, activation of WinStudio licenses and the WinStudio behavior in case of invalid or insufficient licenses
- [chapter 5 "Development Environment" on page 49](#): Describes the "WinStudio" user interface (or "development environment") and basic features and technologies that have to be understood before creating new projects
- [chapter 6 "WinStudio Projects" on page 89](#): Provides step-by-step instructions for creating and configuring new projects
- [chapter 7 "Working with Tags" on page 105](#): Describes basic concepts about the project database, tag types (arrays, classes and pointers), tag values and parameters. Besides conceptual considerations, creating and processing of tags in projects is described
- [chapter 8 "Screens and Graphics" on page 155](#): Describes the development tools of "WinStudio" for creating individual images and graphics
- [chapter 9 "Alarms, Events and Trends" on page 247](#): Describes the application of alarm and trend tasks to log historical data and the use of the objects alarm/event and trend for display of historical data in a project image
- [chapter 10 "Background Tasks" on page 315](#): Describes the creation and configuration of different "tasks" of "WinStudio" for projects
- [chapter 11 "Communication with Other Devices" on page 333](#): Describes the configuration of projects for reading and writing process data

About this Documentation

and provides instructions for driver configuration and OPC, TCP/IP and DDE communication

- [chapter 12 "Project Security" on page 357](#): Describes the set-up and management of safety systems for projects
- [chapter 13 "Translation of User Texts" on page 381](#): Describes the translation of user texts with the support of the WinStudio translation tool
- [chapter 14 "Testing and Debugging Projects" on page 385](#): Describes the test and troubleshooting process for projects by means of tools like the Database Spy and Output windows.
- [chapter 15 "Provision as a Web Application" on page 391](#): Describes configuration and execution of projects as Web application
- [chapter 16 "Downloading to a Remote Device" on page 425](#): Describes transmission, monitoring and troubleshooting in projects via remote access
- [chapter 17 "Database Interface" on page 429](#): Describes the connection of databases that are compatible with "WinStudio"
- **Scripting languages**
 - [chapter 19 "Appendix: WinStudio-Internal Scripting Language" on page 471](#): Describes all functions of the internal WinStudio scripting language. Each functional description includes complete syntax, possible output values, and examples of usage
 - [chapter 20 "Appendix: VBScript Language" on page 715](#): "WinStudio" implements Visual Basic Scripting Edition 5.5 or higher

A overview of the use of VBScript commands in "Rexroth WinStudio" is provided. Additionally, the most common features of the language are provided as a quick reference. For a full description of the language as well as its interfaces and functions, please refer to [Microsoft Developer Network](#)
- [chapter 18 "Troubleshooting and FAQs" on page 461](#): Provides instructions for verifying projects, describes some common development errors and explains how to contact a support representative

1.3 Using the Safety Instructions

1.3.1 Safety Instructions – Structure

The safety instructions are structured as follows:

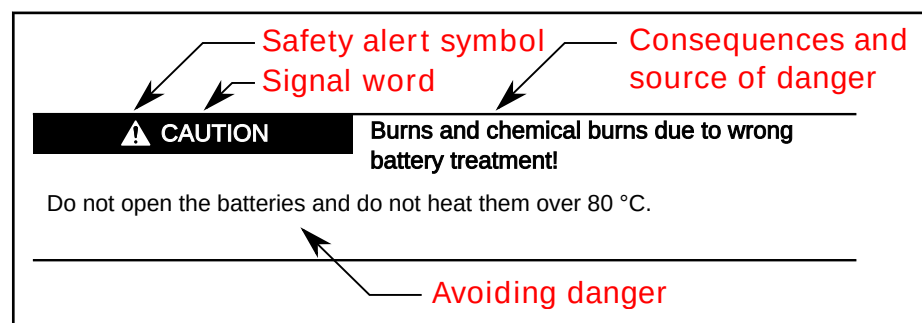


Fig. 1-1: Safety instructions – Structure

1.3.2 Explaining Signal Words and Safety Alert Symbol

The safety instructions in this documentation contain specific signal words (danger, warning, caution, notice) and, if necessary, a safety alert symbol (according to ANSI Z535.6-2006).

The signal word is meant to draw the reader's attention to the safety instruction and signifies the degree of danger.

The safety alert symbol (a triangle with an exclamation point), which precedes the signal words danger, warning and caution is used to alert the reader to personal injury hazards.

DANGER

In case of non-compliance with this safety instruction, death or serious injury **will** occur.

WARNING

In case of non-compliance with this safety instruction, death or serious injury **can** occur.

CAUTION

In case of non-compliance with this safety instruction, minor or moderate injury could occur.

NOTICE

In case of non-compliance with this safety instruction, property damage could occur.

1.3.3 Symbols Used

Notes are displayed as follows:



This is a note.

Tips are displayed as follows:



This is a tip.

1.4 Conventions Used in this Documentation

In this documentation, standardized formatting and terminology is used for easier understanding.

Text conventions

In this documentation, special text formatting is used to support quick identification of certain terms:

- Titles, names, new terms, and messages are indicated using "mono-space" text (for example, "Object Properties").
- File names, screen text, and text that has to be entered are indicated using a **bold** typeface (for example, **D:\Setup.exe**).

About this Documentation

- Buttons, menu options, and keyboard keys are indicated using a **bold** typeface (for example, **File** menu).
- Mouse and selection conventions** In this documentation, the following mouse and selection conventions are used:
- A PC mouse configured for right-handed use is described, so that the left mouse button is the primary button and the right mouse button is the secondary button.
- **Click** and **Select** both mean to click once on an element with the left mouse button. In general, "click" on buttons and "select" from menus.
 - **Double-click** means to quickly click twice on an element with the left mouse button.
 - **Right-click** means to click once on an element with the right mouse button.
 - **Select** also refers to using the cursor to highlight or specify an element on the computer screen.
- Selecting an element with a touchscreen is usually the same as selecting with a mouse, except that screen objects or sections are touched (selected) with the finger.
- To select elements by keyboard, the Tab key is typically used to move around options, the Enter key to open menus, and the <Alt> key with a letter key to select an object that has an underlined letter.
- **Drag** refers to pressing down the respective mouse button and move the mouse before releasing the button. Usually an outline of the element moves with the mouse cursor.
- Windows conventions** In this documentation, the following Windows conventions are used:
- **Dialogs** are windows that enable configuration of settings and entering information.
 - **Text fields** are areas in dialogs where text can be typed in.
 - **Radio buttons** are white circles in which a black dot appears or disappears when it is clicked on.
- The dot indicates that the option or function is "enabled" (selected). No dot indicates the option or function is "disabled" (not selected).
- **Checkboxes** are white squares in which a check appears or disappears when it is clicked on with the cursor.
- The check indicates that the option or function is "enabled" (selected). No check indicates that the option or function is "disabled" (not selected).
- **Buttons** are icons in boxes that appear "pressed" when they are clicked on.
 - **Lists** are panes (white boxes) in windows or dialogs containing two or more selectable options.
 - **Combo boxes** have an arrow that, when clicked, displays a part or all of an otherwise concealed list.
 - **Dockable windows** are windows that can be dragged to an edge of the user interface and dock to that edge.

1.5 Required and Supplementing Documentation

In addition to this technical reference, consult the following manuals, if necessary:

About this Documentation

- "First Steps Manual": Provides basic information about "Rexroth WinStudio", including a systematic tutorial that allows developing a single project and becoming familiar with the product in a very short time.
- "Drivers User Manual": Explain how to configure individual direct communication drivers, according to their unique protocol characteristics. One customized user manual is included with each driver.



All manuals are located in the "Documentation" folder on the "WinStudio" installation CD.

"WinStudio" installs the Drivers User Manuals "BR_WS.pdf" in the \Drv or \Drive-DE folder in the program directory.

Technical information can also be accessed from the **Help** menu:

IndraWorks Online Help ► IndraWorks components ► HMI tools ► Bosch Rexroth WinStudio driver

2 About this Application

2.1 About this Application, General Information

"Rexroth WinStudio" (or simply "WinStudio" or Studio) is a powerful, fully integrated software program that enables designing and creating "HMI" (Human-Machine Interface) or "SCADA" (Supervisory Control and Data Acquisition) projects for:

- Data acquisition
- Local supervisory stations
- Remote supervisory stations
- Data concentration on distributed processes
- Data communications with corporate systems

2.2 Product Overview

"Rexroth WinStudio" projects run on visualization devices (e.g. IndraControlV) connected in real-time to machines or processors through programmable controls, remote I/O devices, and other data-acquisition equipment.

These projects consist of animated images, configurable PLC (programmable logic control) drivers and other controllable I/O devices, a project tags database, and optional modules such as alarm monitoring, logic, trend charts, recipes, schedulers, and a security system. "WinStudio" projects communicate with industrial I/O systems and other Windows applications in the run-time environment using the following protocols:

- ODBC ("Open Database Connectivity")
- DDE ("Dynamic Data Exchange")
- NetDDE ("Network""Dynamic Data Exchange")
- OPC ("Open Connectivity")
- TCP/IP ("Transmission Control Protocol/Internet Protocol")

After a project is created, it can be executed in the development environment or (via a TCP/IP connection) downloaded to the target device and executed with the "Rexroth WinStudio" runtime software.

The processes of the runtime environment communicate with the connected devices and process or visualize the projected data.

The product consists of two parts:

- Development environment: The WinStudio development environment is part of IndraWorks Engineering and can be run on desktop PCs, notebooks or industrial PCs if a supported Microsoft Windows operating system for desktop or server is installed.
- Runtime system software: Operable on visualization devices (e.g. IndraControlV) if a currently supported Microsoft Windows desktop operating system or Windows Embedded is installed.



The runtime client for Windows CE operating systems (CEView) is often pre-installed on the HMI.

If necessary, the CE View version can be updated by downloading the current version from the development computer to the HMI.

About this Application

2.3 Product Features

"Rexroth WinStudio" provides the following features:

- [Development Environment on page 49](#) integrated in IndraWorks Engineering with editors, toolbars, dialogs and menus:
 - Customizable toolbars
 - Tasks, objects and controls displayed in the Project Explorer [Project Explorer on page 55](#)
- Versatilely configurable objects and animations (the ability to modify object properties, execute commands or write values to tags used to create display screens at application runtime):
 - Configurable objects such as buttons, rectangles, ellipses, polygons, lines and text
 - [Object animations on page 220](#) such as bar graphs, colors, resizing, change in position, hiding/showing, rotation, command execution, hyperlink and text input/output
 - Online and historical display of [alarm lists on page 247](#)
 - Online and historical [trends on page 269](#)
 - The [Formatting Screen Objects on page 233](#) is supported by the toolbars "Alignment functions" and "Object properties". Tools are provided to arrange, reposition and resize selected screen objects as well as to modify the objects' styles and fonts.
 - [Background Color or Picture of a WinStudio Screen on page 160](#) can be edited or created.
 - Graphics import
 - Containers for [ActiveX objects on page 208](#) and .Net controls
- Remote management and configuration of the runtime environment
- Microsoft DNA architecture compliance, with full OPC and XML support
- Activation of a web interface to convert images to an internet-capable format and to display them by a client (web-browser). Data is exchanged online via the TCP/IP protocol
- [Symbols library on page 199](#) with more than 100 available objects, such as pushbuttons, measuring instruments, sliders, switches, text and numeric displays, LED displays, lines, pumps, icons, vehicles, valves, frames, motors, gauges and controls
- Troubleshooting tools:
 - [Database Spy window on page 84](#) to monitor/set tag values and execute functions
 - [LogWin module on page 85](#) to record OPC, DDE, and TCP/IP protocols, to activate modules, trace tags, etc.
 - Cross reference search for tags within a WinStudio application
 - System and network diagnostics at runtime
- Powerful and flexible tag database (Boolean, Integer, Real, and String tags), array tags, classes, and indirect tags (pointers)
- Open architecture with API, exchange of tag values with external software
- [Translation editor on page 381](#) for translation of a project to various languages and switching between the languages during the project runtime

About this Application

- Client and server communication based on [TCP/IP on page 351](#) to exchange tag values and configure redundant systems
- [Drivers for direct communication on page 333](#) to connect to Rexroth controls (BR_WS)
- OPC server and OPC client with integrated [OPC Browser on page 342](#)
- Runtime [security on page 357](#) with password protection for images and objects (256 levels)
- Logical expressions and a [scripting language on page 471](#) with more than 200 functions
- Tools to create [recipes on page 319](#) and [reports on page 320](#) (in ASCII, UNICODE and RTF format)
- [Scheduler on page 326](#) for events based on date, time or data conditions (100 ms resolution)
- Import/export function for easy implementation of images and worksheets in other projects.
- [Dial-up functions on page 471](#) to establish, monitor and disconnect dial-up connections with the RAS server of remote stations
- Functions to [send e-mails on page 471](#) from "WinStudio" (or CEView)

2.4 How the Software Works

2.4.1 Internal Structure and Data Flow

The "WinStudio" runtime environment is executed on a visualization device (Windows 2K/XP/Vista/7/CE) and consists of the following modules or "Threads" (program elements that can be executed independently of each other):

- **Background Task** (supervisory task): Executes other internal tasks ("WinStudio" worksheets). For example, the "Background" Task executes scripts configured in the [Math on page 324](#) and [Scheduler on page 326](#) worksheets and manages parameters configured in the [Alarm on page 247](#), [Recipe on page 319](#), [Report on page 320](#) and [Trend on page 269](#) worksheets.
- **Database Spy** (error analysis tool):
 - Executes functions and/or expressions for testing purposes
 - Reads data (such as tag values) from the "Tags" database
 - Writes data (such as tag values) to the "Tags" database
- **DDE Client**: Manages the DDE communication with a DDE Server (local or remote), according to parameters configured in the [DDE Client worksheets on page 352](#).
- **DDE Server**: Manages DDE communication with a DDE Client (local or remote).
- **LogWin** (error analysis tool): Records messages generated from other modules/tasks.
- **Driver Runtime**: Manages the read/write commands configured in the [Driver worksheets on page 333](#).
- **OPC-Client (OPC DA and OPC UA)**: Manages OPC communication with an OPC Server (local or remote), according to parameters configured in the [OPC Client worksheets on page 342](#).
- **OPC DA Server**: Manages OPC DA communication with an OPC DA Client (local or remote)

About this Application

- **ODBC Runtime:** Manages ODBC data communication with any SQL relational database, according to parameters configured in the [ODBC worksheets on page 321](#).
- **TCP/IP Client:** Manages TCP/IP communication with a remote TCP/IP server module (from "WinStudio"), according to parameters configured in the [TCP/IP client worksheets on page 351](#).
- **TCP/IP Server:** Manages TCP/IP communication messages with a remote TCP/IP client module (from "WinStudio").
- **Viewer:** Executes all scripts (On Open, On While, On Close, Command, Hyperlink, and so forth) configured for screens and updates the screen objects.

None of the preceding runtime modules exchanges data directly with another module or task data. Instead, the runtime modules transmit data to the "Tags" database or receive data from there. This can be referred to as the "heart" of "WinStudio".

The "Tags" database manages the flow of data between modules. In addition, the "Tags" database stores all tag values and the status of all properties associated with each tag (such as alarm conditioning, time stamp, quality, and so forth).



Tags are variables (such as communication points to field equipment, calculation results, alarm points, and so forth) that are used in screens and worksheets.

For detailed information about tags, tag values, and tag properties, refer to [Working with Tags on page Working with Tags](#).

About this Application

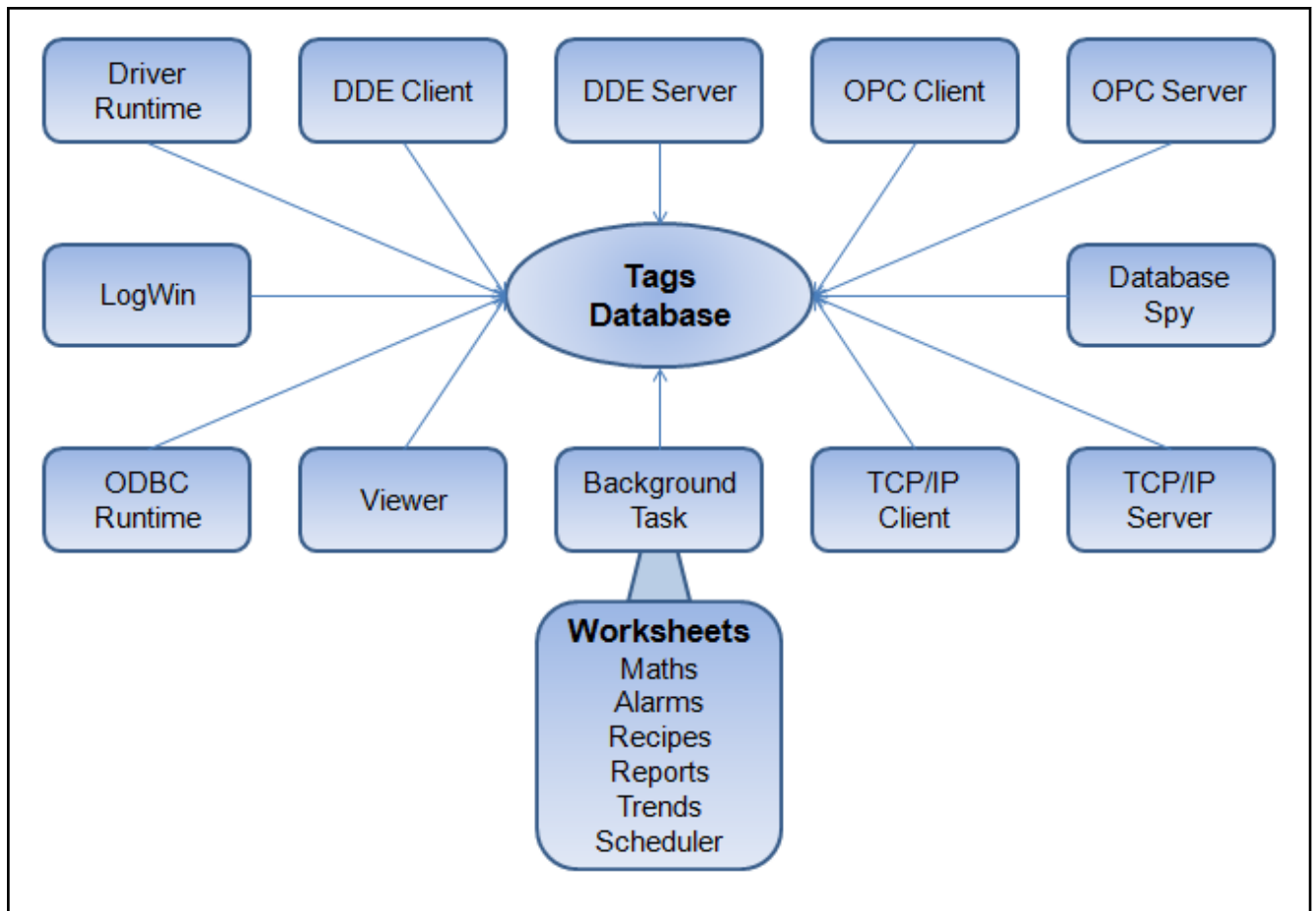


Fig.2-1: Data flow

Each "WinStudio" module contains a virtual table of the tags that are currently relevant for that module. The "Tags" database uses this table to determine which information has to be updated in each module. For example, the "Viewer" contains a virtual table that lists all tags for all of the open WinStudio screens. If a tag value changes, the "Tags" database sends a message to the "Viewer" that updates the value in all objects.

For example, if a driver reads out a new value from the PLC, the driver updates the tag in the "Tags" database.

If this new information has to be displayed on the WinStudio screen, the "Tags" database sends the new tag value to the "Viewer" task, and the "Viewer" updates the screen.

About this Application

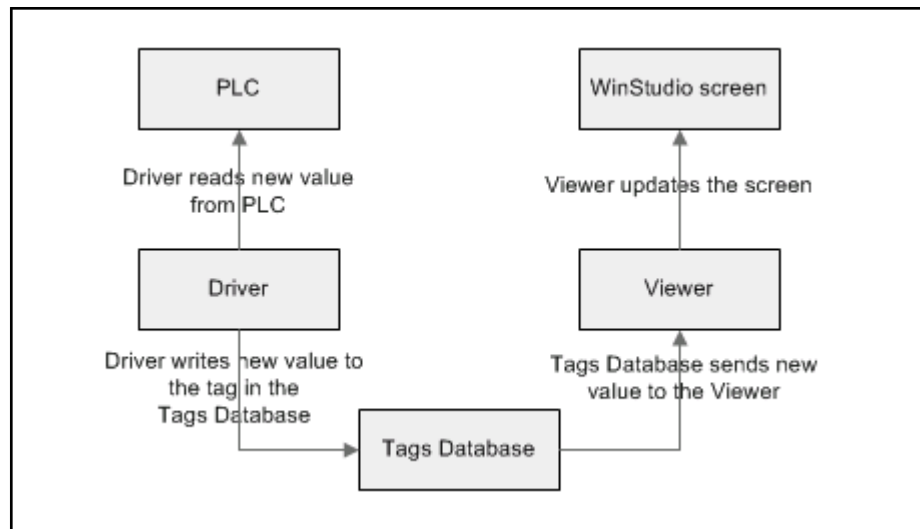


Fig.2-2: Data flow example

Note that the driver does not send new tag values directly to the "Viewer". In addition, there is no polling between tasks – the "Tags" database receives the updated information and immediately forwards it to all runtime tasks requiring that information.



The "Runtime" module only updates an object when (at least) one of the object's tag values change.

If an object animation (such as [Text Data Link on page 225](#)) is configured with a function that does not require a tag (for example, [NoInputTime on page 625](#)), the "Viewer" does not update the object.

The architecture of "WinStudio" significantly improves the internal data flow performance and makes it easy to add new internal tasks. Even though each task works independently, it can access information from any other task through the "Tags" database.

2.4.2 Executing/Switching Modules

"WinStudio" is a SCADA system composed of modules that have to be executed simultaneously. Based on the multi-tasking concept, each runtime module ("Viewer", "Driver", etc.) is a "thread" and the operating system automatically switches from one thread to another. It is a common misconception that a SCADA system is executed simultaneously with a PLC program. A PLC program contains a simple loop:

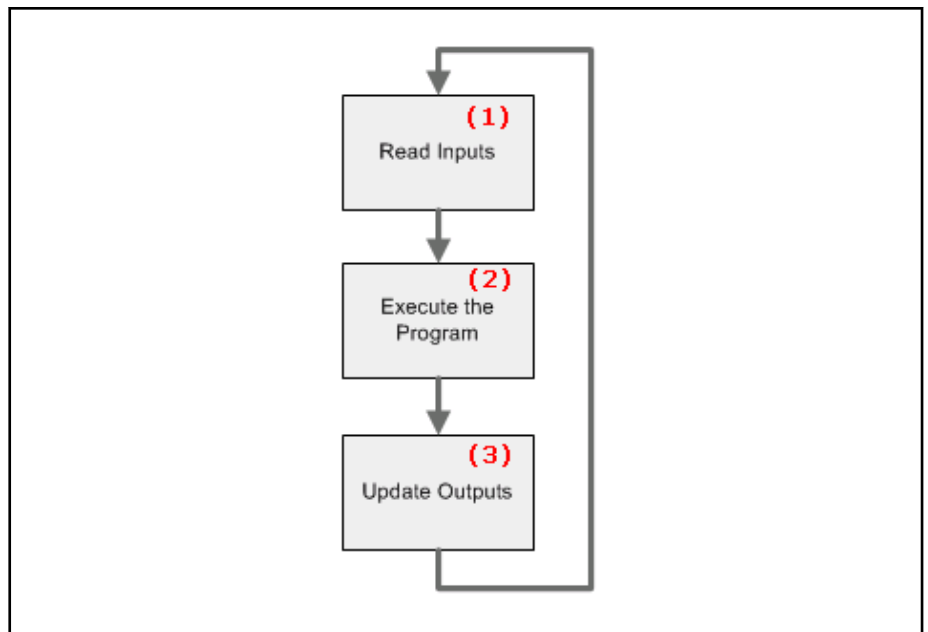


Fig. 2-3: PLC program loop

However, in a SCADA system, there are several modules running simultaneously, most of which can read and write data.

As a SCADA system continuously modifies data (tag values) during task execution, the preceding diagram is "not" applicable.

"WinStudio" is only executed in a single process: "Studio Manager.exe". When a runtime project is executed, the "Studio Manager.exe" process starts the "Tags" database and all of the runtime modules configured for the project. It can be specified which modules (such as "Viewer" and "Driver") start during the runtime.

Each Windows process lists "active" threads for the operating system. Actually, each process activates and deactivates each thread during the runtime, according to the algorithm of each process. Also, specify a priority value when starting a thread. The operating system continuously scans all currently active threads and executes the threads according to their priority value - executing the higher-priority threads first. When threads with higher-priority values are active, the threads with lower-priority values are not executed at all. If there is more than one thread with the same priority value, and there are no other threads with higher-priority values, the operating system keeps switching between the threads with the same priority.

All "WinStudio" threads are set to priority "7", which is "THREAD_PRIORITY_NORMAL". (Most programs contain this priority value.)

Real-time program (such as SoftPLCs and Device Drivers) threads are assigned a higher-priority value ("THREAD_PRIORITY_HIGHEST"); however, these programs have to provide a mechanism to keep them inactive for some period of time, otherwise the threads with normal priority are never executed.

"WinStudio" uses the "UNICOMM.DLL" library for serial drivers.

This library creates a "THREAD_PRIORITY_HIGHEST" thread that "sleeps" (remains inactive) until data arrives in the serial channel.

When "WinStudio" detects new data in the serial channel, the "THREAD_PRIORITY_HIGHEST" thread "wakes up" (becomes active) and transfers the data from the operating system buffer to the thread buffer, where it can be read by the Driver. This thread is the only highest-priority thread created by "WinStudio".

About this Application

If threads are allowed to remain active all the time, the CPU usage is 100% all the time, which has to be avoided for performance reasons. Every program provides a mechanism to prevent threads from staying active permanently.

"WinStudio" uses the following parameters to prevent that threads permanently remain active:

- **TimeSlice** (from operating system): Causes the operating system to switch automatically between active threads with the same priority value. By default, the operating system executes each active thread for approximately 20 ms and then switches to the next active thread. This means that no active thread is executed longer than 20 ms if there are multiple threads with the same priority value.
- **TimeSlice** (from "WinStudio"): Specifies how long each "WinStudio" thread can remain continuously active.

This parameter is used in addition to the operating system's TimeSlice parameter. A TimeSlice value is configured for each "WinStudio" thread (except the Background Task) and specify how long each thread can remain continuously active. As long as a thread is active, the operating system can switch to that thread.

- **Period** (from "WinStudio"): Specifies the maximum amount of time each "WinStudio" thread (except the Background Task) can remain inactive.

NOTICE

It is "highly" recommended not to modify these default values unless it is absolutely necessary. Configuring any of these parameters incorrectly can cause the entire system to malfunction (for example, CPU load rises to 100 %) and/or cause some tasks to perform poorly.

If the parameter defaults are to be changed, take a note of the values before changes are made to be able to return to the original settings in case of a malfunction.

To change the default values for the "WinStudio" parameters TimeSlice and Period, open the file "Program Settings.INI" in folder "<IndraWorks directory>\WinStudio\bin" with a text editor.

The following is a list of all parameters contained in this .ini file and their default values (in milliseconds).

Program Settings.INI

```
[Period]
DBSpy=1000
UnidDEClient=200
UnidDE=200
Driver=20
LogWin=100
UnioDBCRT=100
OPCClient=20
OPCServer=20
TCPClient=100
TCPServer=100
Viewer=50
```

```
[TimeSlice]
UnidDEClient=100
Driver=10
OPCClient=10
OPCServer=10
TCPClient=200
```


About this Application

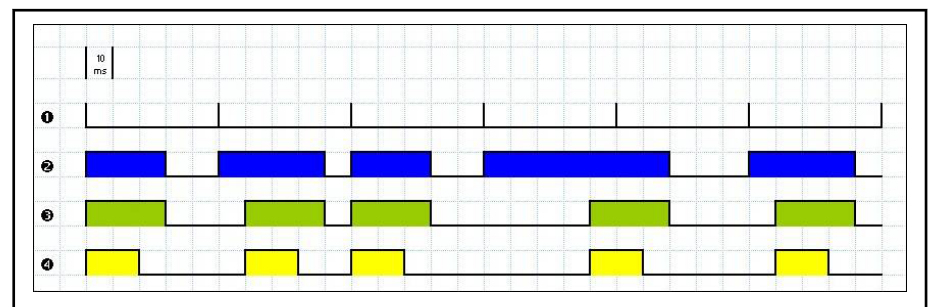
TCPServer=200
Viewer=200



It may be the case that not all listed parameters can be found in the file "Program Settings.ini". Nevertheless, "WinStudio" uses parameters not included and assigns default values to these parameters.

- To change the default values of displayed parameters, overwrite the current value and save the file.
- If a parameter value is to be changed that is not included in the file, add the parameter name and the desired value as illustrated in [Program Settings.INI on page 32](#).

The following figure illustrates how "WinStudio" executes a generic thread (such as the "Viewer").



10 ms Time period of the diagram
Fig.2-4: Executing a generic thread

The following apply here:

- Signal 1 is the "Period" time period (set to 50 ms for this example)
- Signal 2 displays when the thread for the operating system is active
- Signal 3 is the "TimeSlice" time period (set to 30 ms for this example)
- Signal 4 shows the execution of the thread itself

In this example, "WinStudio" generates a Period message every 50 ms (signal 1). When "WinStudio" generates this message, its thread becomes active and remains active until the specified TimeSlice time period (from "WinStudio") expires. The thread then remains inactive until "WinStudio" generates the next Period message (signal 1).

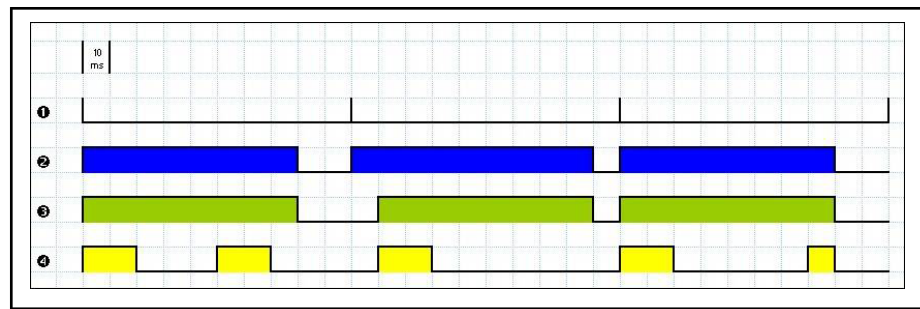
While the thread is active, the operating system is responsible for executing that thread. A thread being active does not mean that the operating system executes it immediately. For example, this is the case if the operating system is currently executing different threads.

When the operating system executes the thread, the "TimeSlice" timer starts counting and the thread is executed for 20 ms (TimeSlice of the operating system). After the 20 ms period, the operating system automatically switches to the next active thread (such as the driver), and so on.

In the above example, the TimeSlice time was set to 30 ms, which means the operating system is not supposed to execute the thread more than once in each "WinStudio" TimeSlice. However, if higher values are specified for the "WinStudio" TimeSlice time period, it is likely that the operating system executes the same thread more than once in the same TimeSlice time period.

In the next example, the Period and the TimeSlice values were changed as follows, but the default operating system TimeSlice period (20 ms) was not changed.

About this Application



10 ms Time period of the diagram

Fig.2-5: Setting a higher TimeSlice

The following apply here:

- Signal 1 is the Period time period (set to 100 ms)
- Signal 2 displays when the thread for the operating system is active
- Signal 3 is the "WinStudio" TimeSlice time period (set to 80 ms)
- Signal 4 shows the execution of the thread itself

Please note that the thread can be executed more than once in the same TimeSlice time period. When the "WinStudio" TimeSlice time period expires, the operating system interrupts the thread execution; however, even though the "WinStudio" Period and TimeSlice parameters are set to 100 ms and 80 ms respectively, the operating system does not execute this thread continuously for more than 20 ms, because the operating system TimeSlice time period is set to 20 ms.

If the operating system is not executing the "Viewer" thread, the CPU can execute any other thread or remain inactive (if there are no other active threads to execute). Remember: The "WinStudio" Period and TimeSlice parameters were created to prevent all threads from being active at the same time to prevent 100 % CPU usage.

During thread execution, the thread has to handle its pending messages. For example, the Viewer module has to update any related screen objects. If there are no messages pending, the thread deactivates itself and gives control back to the operating system. The operating system immediately switches to the next active thread. In other words, a thread can interrupt its own execution – even if the operating system TimeSlice time period has not yet expired (which occurs frequently in real-world applications).



The "Database Spy", "DDE Server", "LogWin", and "ODBC Runtime" modules do not support a TimeSlice parameter. Thus, the thread becomes inactive after processing all pending messages until the next Period message for the respective thread occurs.

The Background Task is an exception of the described thread execution mechanism. The execution of the Background Task is described in the following.

2.4.3 Executing/Switching the Background Task

The Background Task executes scripts from the "Math" and "Scheduler" worksheets (for example, messages from "Alarm" and "Trend" worksheets). In addition, the Background Task executes all recipe and report commands when the **Recipe()** or **Report()** functions are executed during runtime.

Although the "Alarm", "Math", "Scheduler", and "Trend" tasks are no threads, their Period time can be specified or changed in the **Program Settings.ini** file located in the "WinStudio" program directory.

About this Application

The Period default values (in milliseconds) are as follows:

Program Settings.INI

```
[Period]
Math=100
Sched=50
Alarm=100
Trend=1000
```

These values mean that every 100 ms, "WinStudio" generates a Period message to the Math task. Every 50 ms, "WinStudio" generates a Period message and sends it to the Scheduler task, and so on.

NOTICE

It is highly recommended that Background Task default values are not modified unless it is absolutely necessary. Configuring any of these parameters incorrectly can cause the entire system to malfunction (for example, CPU usage rises to 100%) and/or cause some tasks to perform poorly.

If the parameter default values are to be changed, take note of the values to be able to return to the original settings in case of a malfunction.

Keep in mind that the Background Task thread has the same priority as other threads in "WinStudio" ("Drivers", "Viewer", etc.), which means that the operating system does not execute this task continuously for more than 20 ms.

The Background Task executes the [Recipe on page 319](#) and [Report on page 320](#) worksheets when the [Recipe on page 558](#) and [Report on page 558](#) functions are called. As the "Recipe()" and "Report()" functions are "synchronous", once the Background Task starts executing the functions, it does not switch to another task ("Alarm", "Math", "Scheduler" or "Trend") until it has executed the functions completely. Executing a **Recipe()** or **Report()** function usually takes a few milliseconds.

The Background Task has to switch between the "Alarm", "Math", "Scheduler", and "Trend" tasks. When the Background Task switches to the "Scheduler" task, it does not switch to another task ("Alarm", "Math", or "Trend") until all "Scheduler" worksheets are processed. After processing all "Scheduler" worksheets, the Background Task does not execute the "Scheduler" again until it receives the next Period message for the "Scheduler" task.

The Background Task applies the same behavior when executing the "Alarm" and "Trend" tasks — when the Background Task switches to the "Alarm" or "Trend" task, it does not switch to another task until all pending messages have been processed. The Background Task does not execute the "Alarm" or "Trend" task again until "WinStudio" generates the next Period message for each of these tasks.

The Background Task typically executes the "Alarm", "Scheduler", and "Trend" tasks in a few milliseconds. However, it can take longer to execute the "Math" task because it usually contains loops and complex scripts. Consequently, the mechanism used to execute the "Alarm", "Scheduler", and "Trend" tasks cannot be applied to the "Math" task.

The Background Task executes the "Math" task for no more than 10 ms continuously before switching to another task (such as the "Scheduler"). The Background Task cannot execute the "Math" task again for the next 50 ms; however, the Background Task can execute other tasks ("Alarm", "Recipe", "Report", "Scheduler", or "Trend") during this 50 ms period. After the Background Task has executed all of the "math" worksheets, it does not begin a

About this Application

new scan of the "math" worksheets until "WinStudio" generated a new Period message for the "math" task.

It is important to re-emphasize that this process was created to prevent 100 % CPU usage all the time.

NOTICE

Be sure to be careful when using the [Math on page 557](#) function in a "Scheduler" worksheet or for a screen object (such as animation).

If the "Scheduler" task executes a **Math()** function, no other task can be executed by the Background Task until the "Scheduler" has executed the entire "math" worksheet called by the **Math()** function. This process can take several milliseconds or even seconds, depending on the configuration of the script in the "math" worksheet (especially for loops).

If a **Math()** function for a screen object is configured, the "Viewer" stops updating the screen until the "Viewer" has executed the entire "math" worksheet called by the **Math()** function.

If the **Math()** function has to be used for the "Scheduler" task or a screen object, proceed as follows to prevent process delays: By implementation of an auxiliary tag, it can be prevented that the Background Task does not execute the "math" worksheet on the respectively next scan before the auxiliary tag was reset to the value 1.

1. Create an auxiliary tag and enter its name into the "Execution" field of the "math" worksheet to be executed. When the Background Task scans the "math" worksheet, "WinStudio" executes the worksheet
2. Reset the tag in the last line of the "math" worksheet to the value 0
3. In the Scheduler or the screen object, set the auxiliary tag value to 1. The Background Task then executes the "math" worksheet once. Only if the auxiliary tag value is reset to 1, the Background Task again executes the "math" worksheet

3 Installation

3.1 Installation, General Information

This section includes installation instructions for "Rexroth WinStudio", Engineering, EmbeddedView and CEView.

3.2 System Requirements

In the following, the minimum requirements to install and run the "Rexroth WinStudio" software are described.

Development / Project server / Project client

The following is required to install and run the full "Rexroth WinStudio" software:

- A Windows-compatible computer or Windows Embedded-compatible device with a standard keyboard, mouse, and SVGA-minimum display
- A Windows or Windows Embedded operating system that is currently supported by Microsoft, which at this time includes:
 - Microsoft Windows XP Service Pack 3 or later
 - Microsoft Windows 7, all versions
 - Microsoft Windows XP Embedded Service Pack 3
 - Microsoft Windows Embedded Standard 7 (2009)
- Microsoft Internet Explorer 6.0 or later
- Minimum of 500 MB free hard drive space
- Ethernet adapter or wireless networking
- CD-ROM drive (optional, to install the application; it can also be downloaded from our website)
- USB port (optional, to be used with hard key licensing)
- Serial COM ports and adapters (optional, to be used for direct communication with PLCs and other devices)

The full "Rexroth WinStudio" software cannot be installed on Windows Embedded Compact devices.

Project server – Embedded

The following is required to run a device as a project server (i.e., to host the project runtime) using EmbeddedView or CEView:

- A Windows Embedded-compatible device
- A Windows Embedded operating system that is currently supported by Microsoft, which at this time includes:
 - Microsoft Windows XP Embedded Service Pack 3
 - Microsoft Windows Embedded Standard 7 (2009)
 - Microsoft Windows Embedded Compact (formerly known as Windows CE or Windows Mobile), version 5.0 or later
- Minimum of 500 MB free hard drive space
- Ethernet adapter or wireless networking
- USB port (optional, to be used with hard key licensing)
- Serial COM ports and adapters (optional, to be used for direct communication with PLCs and other devices)

Project client – Embedded

The following is required to run a device as a project client using EmbeddedView or CEView:

Installation

- A Windows Embedded-compatible device
 - A Windows Embedded operating system that is currently supported by Microsoft, which at this time includes:
 - Microsoft Windows XP Embedded Service Pack 3
 - Microsoft Windows Embedded Standard 7 (2009)
 - Microsoft Windows Embedded Compact (formerly known as Windows CE or Windows Mobile), version 5.0 or later
 - Ethernet adapter or wireless networking
- Project client – Thin** The following is required to run a device as a project client using either the Secure Viewer program or the browser-based Thin Client:
- A Windows-compatible computer or Windows Embedded-compatible device
 - A Windows desktop, server, or embedded operating system that is currently supported by Microsoft, which at this time includes:
 - Microsoft Windows XP Service Pack 3 or later
 - Microsoft Windows Vista Service Pack 1 or later
 - Microsoft Windows 7, all versions
 - Microsoft Windows Server 2003 Service Pack 2 or later
 - Microsoft Windows Server 2008, all versions
 - Microsoft Windows XP Embedded Service Pack 3
 - Microsoft Windows Embedded Standard 7 (2009)
 - Microsoft Windows Embedded Compact (formerly known as Windows CE or Windows Mobile), version 5.0 or later
 - Microsoft Internet Explorer 6.0 or later
 - Ethernet adapter or wireless networking

3.3 Installing the Software

Engineering

The "Rexroth WinStudio" development environment is integrated in IndraWorks Engineering.

The application for software development can be installed from the "Rexroth IndraMotion MLC/XLC" drive.

Runtime

During installation of the runtime environment on the target device, a distinction has to be made according to the device type and the application type:

- For execution of "HMI standard applications" on VPx or VEx devices with Windows XP/Windows7 or Windows XP Embedded, "IndraWorks Operation" is to be installed on the target device.
- For execution of pure "WinStudio applications" on VPx or VEx devices with Windows XP/Windows7 or Windows XP Embedded, "WinStudio Runtime" is to be installed. Alternatively, IndraWorks Engineering can be applied to establish the connection to the VPx device and to install the system files required for the runtime.
- For execution of "WinStudio applications" on VEx devices with Windows CE, start IndraWorks Engineering, connect to the target device and install the runtime environment from there.



- Administrator privileges are required on the PC in order to install or uninstall the development application.
- Older versions of the development application have to be uninstalled (or moved to a different directory) before installing a new version. Please note also that the same version of the development application cannot be installed in two different directories on the same PC.

Installing CEView on a Windows Embedded device

CEView is the runtime component for "WinStudio" projects on Windows Embedded devices. CEView has to be installed on the device "before" sending a project to it.

Location of the files

Due to the various architectures of Windows Embedded devices, each combination of OS version and device processor has its own runtime environment. The various runtime environments can be found under the following directory:

[...]\WinStudio\Redist\

The particular runtime environment for the specific device can be found under the following directory:

[...]\WinStudio\Redist\<version>\<processor>\

...where:

- **version** is the version of the operating system on the device where CEView is installed:
 - The **WinCE 5.0** folder contains the files for Windows CE and Windows Mobile 5.0 or later.
 - The **WinEmbedded** folder contains the files for Windows XP Embedded and Windows Embedded 7.
- **processor** is the processor used by the specific Windows CE device. WinStudio provides a runtime for every processor that is currently supported by the Windows CE operating system (e.g., Pocket2003-ArmV4, ArmV4i, x86). For more information, refer to the manufacturer's documentation for the respective device.

To install CEView on a Windows Embedded device, proceed as follows:



It is recommended using TCP/IP whenever possible.

1. Make sure that the Windows Embedded device is connected to the development station.
2. Turn on the device.

The "Remote Agent" dialog should open automatically. If this is not the case, the file "CEServer.exe" has to be started manually on the device.
3.
 - On a Windows platform, the "CEServer.exe" file is located in the [...]\WinStudio\Bin folder.
 - On a Windows-CE platform, the file is located in the [...]\WinStudio folder.
4. Execute "CEServer.exe" on the target workstation and click on "Setup" in the **Remote Agent** dialog.

Installation

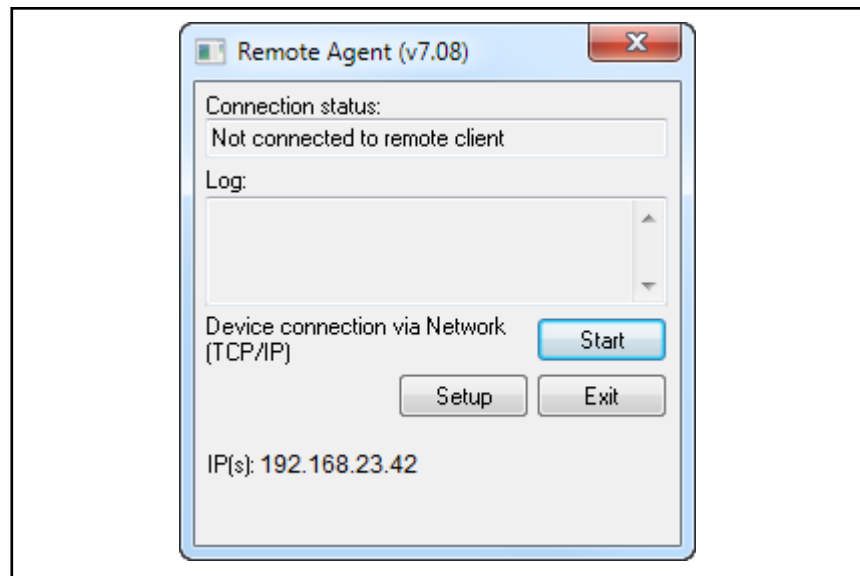


Fig.3-1: Remote Agent dialog

The "Setup" dialog is displayed:

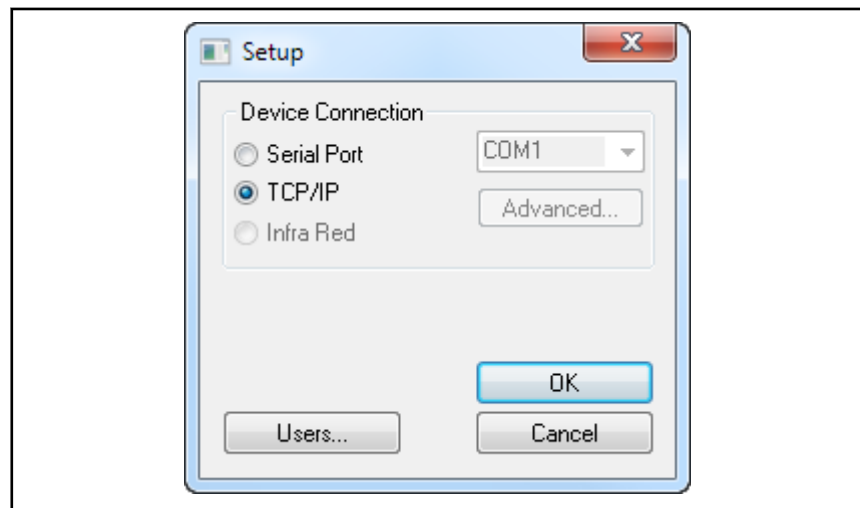


Fig.3-2: Setup dialog

Use the parameters on this dialog to configure communication between the development and target systems:

- **Serial Port:** To establish a serial connection to the project server, select "Serial Port" and then select the specific port from the selection list. Click **Advanced**, to display the "Serial Advanced Settings" dialog.

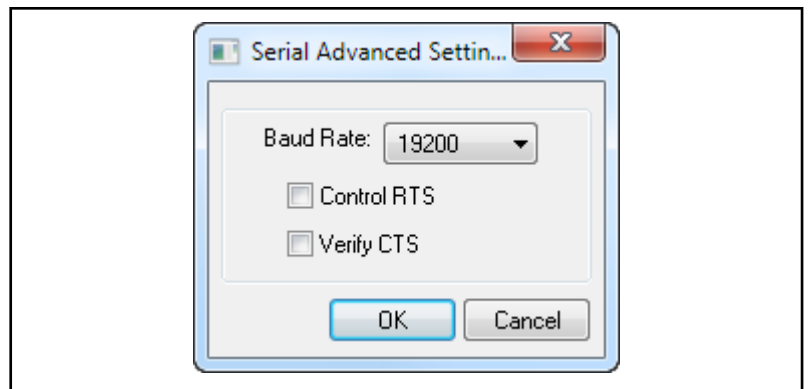


Fig.3-3: Advanced Settings dialog

The parameters on this dialog can be used to control the flow of data between target and development stations: After adjusting these parameters, click **OK** to close the "Serial Advanced Settings" dialog.

- **TCP/IP:** Enable this option to establish the connection to the development station via TCP/IP.



For better performance, it is recommended to use a TCP/IP connection instead of a Serial Port connection.

5. To close the "Setup" dialog, click on **OK**. Leave the Remote Agent program running in the remote workstation.

After configuration of the target system to receive data, proceed as follows to transfer the runtime environment (CEView) from the development station to the target device:

1. Start the development application on the PC.
2. Open a project or create a new one.
3. Add a visualization device "VPx or VEx with Windows XP/Windows7 or Windows XP embedded or a VEx with Windows CE" and select the application type "WinStudio application" in the HMI wizard.
4. Open the context menu of the "WinStudio" element and select the entry **WinStudio target platform**.

The "Remote Management" dialog is displayed.

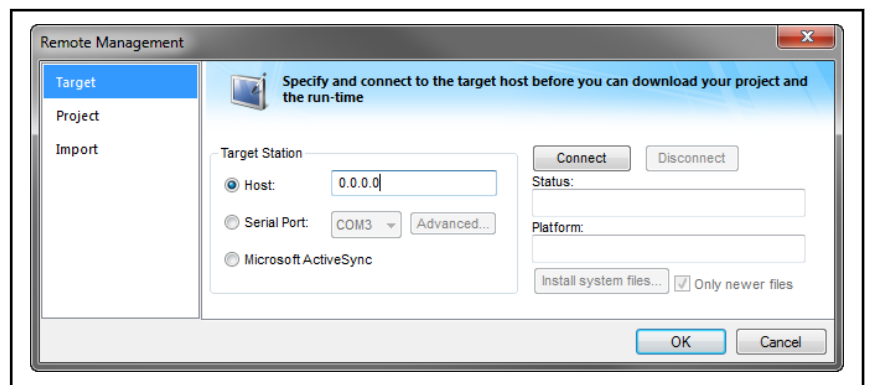


Fig.3-4: Remote Management dialog

Installation

5. Select the **Target** tab and one of the following options to specify the type of connection to the target system.
 - **Host:** Select this option and enter the IP address of the target system into the text box if TCP/IP connection was specified when configuring the target system.
 - **Serial Port:** Select this button and select a port from the combo-box list if a Serial Port connection was specified when configuring the target system.
 - **Microsoft ActiveSync:** Select this button to connect to devices on which ActiveSync is enabled.
6. Click **Connect** to establish a connection between the development application and the operating device.
 In the **Status** text field, the current status of the connection is displayed.
 If the connection is successful, then the device specifications of the connected operating station are displayed in the **Platform** text field.
7. Click on **Install system files...** to transfer the files for the respective run-time environment to the operating station.
8. When the installation is completed, click **Disconnect**.

3.4 Starting the Engineering Software

To run "Rexroth WinStudio", proceed as follows:

- Double-click the "IndraWorks Engineering" icon on the desktop; or
- Select
Start ► All Programs ► Rexroth ► IndraWorks ► Engineering.

Open or create a project with a visualization device.

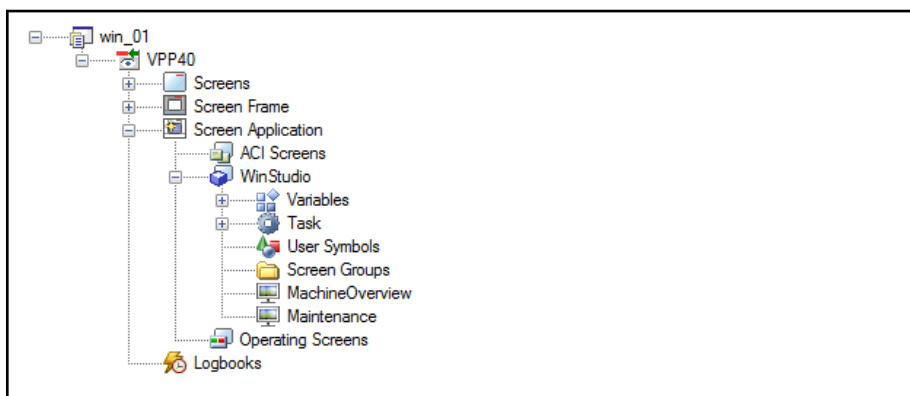


Fig. 3-5: The WinStudio folder

Select an element under the WinStudio folder.

Open the "Rexroth WinStudio" Engineering software by double-clicking the element.



The WinStudio development environment can be run under any display setting.

However, it is recommended to configure the display settings to a resolution of 1024x768 (or higher) and to use more than 256 colors for a more pleasing environment.

The project resolution (screen size) is independent of the operating system resolution.

4 Licensing

4.1 License Protection Types

"Rexroth WinStudio" and CEView support the following license protection types:

Hardkey

An encapsulated chip that has to be physically connected to the USB interface of the computer.

The software license is included in the hardkey and this license cannot be shared simultaneously with more than one other copy of software in the network. If the hardkey is connected to another computer, the license is effectively transferred to that computer.

While using the USB hardkey, the respective USB port cannot be used for any other device.

Softkey

When installing "Rexroth WinStudio" or CEView, the program generates a unique code that is referred to as "Site Code". This Site Code can be sent to the software vendor, who then generates a license code, the "Site Key", to match the respective Site Code.

The Site Key is used to install the "Rexroth WinStudio" or CEView license on the computer or Windows Embedded device.

4.2 License Settings

Both hardkey and softkey licenses set the following parameters:

Version

The overall [version on page 45](#) of the "WinStudio" software, e.g., "7.3". (This does "not" include the service pack, if any.)

Drivers

The number of direct communication [drivers on page 333](#) that can be simultaneously enabled.

Product type

Specifies which features and restrictions are enabled for the application (e.g., the maximum number of tags supported).

Licensing

Execution mode

Specifies one of the following options:

- **Engineering Only:** Can develop a "WinStudio" project and then run it for short-term testing only. This license cannot be used as a long-term run-time license.
- **Runtime Only:** Can run a "WinStudio" project for an unlimited period. This license cannot be used to develop or modify the project.
- **Engineering + Runtime:** Can develop a "WinStudio" project and then run it for an unlimited period.

For more information, please refer to <Execution Modes>

Thin Clients

The number of [Thin Clients on page 413](#) that can connect simultaneously to the server. Each license enables the connection of a single Thin Client. If more than one client are to be used, a special license is required.

Secure Viewers

The number of [Secure Viewer on page 413](#) Clients that can connect simultaneously to the server. Each license enables the connection of a single Secure Viewer Client. If more than one client are to be used, a special license is required.

SMA Clients

The number of <Studio Mobile Access Clients> (SMA Clients) that can connect simultaneously to the server. Each license enables the connection of a single SMA Client. If more than one client are to be used, a special license is required.

4.3 License Types

"Rexroth WinStudio" and CEView support the following license types that vary according to the available functions:

Execution mode	WinStudio	CEView
Demo license	Y	Y
Lite license	Y	Y
Engineering license	Y	N
Runtime license	Y	Y
Engineering and Runtime license	Y	N

Demo license

If the WinStudio demo license is activated, the full scope of functions of WinStudio including the Engineering functions can be used for 30 days. After the test period expires, the last valid license on the computer is active. The test license can only be activated once on each device.

Even without activating the demo license, CEMView allows execution of the WinStudio runtime for 2 hours. Afterwards, the program is closed. However, CEMView can be restarted.

Lite license

The Lite license enables use of most Engineering functions as well as execution of the WinStudio runtime for projects that are created for the "WinStudio Lite" product type. With Lite licenses, max. 500 variables are available in the WinStudio application. Advanced functions like "Alarms", "Trends", "Reports", "Recipes" and "external databases" are not available with Lite licenses.

Engineering license

Enables all Engineering functions for an unlimited time. The following licensing levels are available: 1500, 4000, 64000 or 512000 variables.

This mode also enables execution of the runtime tasks and debugging tools ("Database Spy", "Output" window, and "LogWin" module) for 24 hours continuously. After the 24-hour period these tasks shut down, however, they can be restarted and run for another 24 hours, and so on. This license can only be used for development and testing.

Runtime license

Enables execution of all runtime and debugging tools ("Database Spy", "Output" window and "LogWin" module) for an unlimited time. The following licensing levels are available: 1500, 4000, 64000 or 512000 variables. Engineering functions cannot be used with this license type.

Engineering and Runtime license

Enables all engineering tools, runtime tasks, and debugging tools ("Database Spy", "Output" window, and "LogWin" module) for an unlimited period of time.



The remote management tool (**WinStudio target platform** in the context menu of the "WinStudio" folder) is always available, independent of the execution mode, so that files on remote stations can always be downloaded.

4.4 Product Versions

The version numbers of "Rexroth WinStudio" and CEMView should be identical. It uses the syntax "X.Y+SPWW" (e.g. "Rexroth WinStudio v7.3" and "CEMView v7.3"), where:

- "X" refers to the **Family version**. The Family version changes only when major enhancements are added to the product technologies and concepts.

Licensing

- "Y" refers to the **Sub-version**: The Sub-version changes when minor enhancements and/or new features are added to the product.
- "WW" refers to the **Service Pack**. The Service Pack version changes when add-on packages have to be installed to accomplish the following:
 - Upgrade files for the version previously installed
 - Fix bugs in the product (showstoppers and no-workarounds)
 - Provide minor improvements before releasing the next version of the product

Each Service Pack release supersedes the previous Service Pack release. For example, SP2 includes all the contents of SP1 and all newly upgraded files, bug fixes, and improvements. SP3 includes all the contents of SP2 and all new upgraded files, bug fixes, improvements and so on.

Both "Rexroth WinStudio" and CEView can execute projects built in previous versions of the product. However, older versions of "WinStudio" and CEView cannot execute projects built or modified in newer versions of the product.

For example, version 6.1 projects cannot be executed using "WinStudio" version 6.0, however, version 6.0 projects can be executed with "WinStudio" version 6.1.



Each license is issued for a specific "Family" version and "Sub-version" (X.Y), and the license is valid for that version (including Service Packs) only. The license is not valid for a newer "Family" version or "Sub-version" of the product. Therefore, if a new version of "WinStudio" or CEView is installed, the license has to be upgraded to the new version being installed. If only a Service Pack is installed, the license does not need to be upgraded.

4.5 WinStudio License Activation

The individual "WinStudio Runtime" and "WinStudio Engineering and Runtime" licenses are subject to additional costs and have to be activated by entering a license key.

The personal license key can be found on the license sheet.

To activate a license in IndraWorks Engineering, go to **Tools ► Options ► General Information ► Software Licenses** in the licensing dialog and select "WinStudio".

The activation dialog can be opened by clicking on "Licensing..." and the activation key can be entered.

After the license has been activated, it may be required to restart WinStudio Engineering or IndraWorks Operation.

Saving the license on a dongle

The activation key is only valid for one PC. To be able to use a license on several PCs, the license can be saved on a dongle. The license is valid as long as the dongle is plugged into an USB port of the computer.

Windows CE licenses (VEx) cannot be transferred to a dongle.

License activation on VEx devices

To enable a WinStudio license on a VEx device with a Windows CE operating system, open the licensing dialog on the VEx device in the Windows start menu via **Program ► Rexroth ► WinStudio ► License management** and enter the activation key. The licensing software is already installed on new VEx devices. On VEx devices without pre-installed licensing software, the

IndraWorks visualization software has to be installed first before enabling the WinStudio license.



The IndraWorks license component for Windows CE is already installed on new devices. If an already existing VEx device is to be updated with the new software, the runtime component has to be installed on the VEx device via the context menu of the VEx device in IndraWorks Engineering with the Install software function. The link may not be available. In this case, start the licensing dialog via `\StorageCard\Rexroth\WinStudio\bin\IndraWorks.LicenseImport.CE.exe`.

The licensing component requires ".Net CompactFramework 2.0" so that the firmware may have to be updated.

4.6 Invalid Licenses

When opening a project with the WinStudio editor that has been created on another computer with a higher class WinStudio license, the elements in WinStudio Engineering requiring a license are not activated. This means that these elements can neither be created nor edited in this case.

When starting IndraWorks Operation, a message is displayed informing the user that the available WinStudio license is insufficient for the current project.

No WinStudio screens are displayed in case of an insufficient or missing license.



If the current project still is to be used, activate the demo license. Use the IndraWorks licensing dialog to activate the license (as described above).

5 Development Environment

5.1 The Development Environment, General Information

The "Rexroth WinStudio" development environment is integrated in IndraWorks Engineering. It includes the following components:

- Project Explorer: "WinStudio" folder with subfolders
- Different editors for WinStudio screens, math sheets, etc.
- Toolbars
- "WinStudio Screen Tools" tool window
- "WinStudio library" tool window
- "LogWin" output window
- "Winstudio Cross Reference Search" output window
- "Winstudio Database Spy" output window
- "Winstudio Symbols" output window

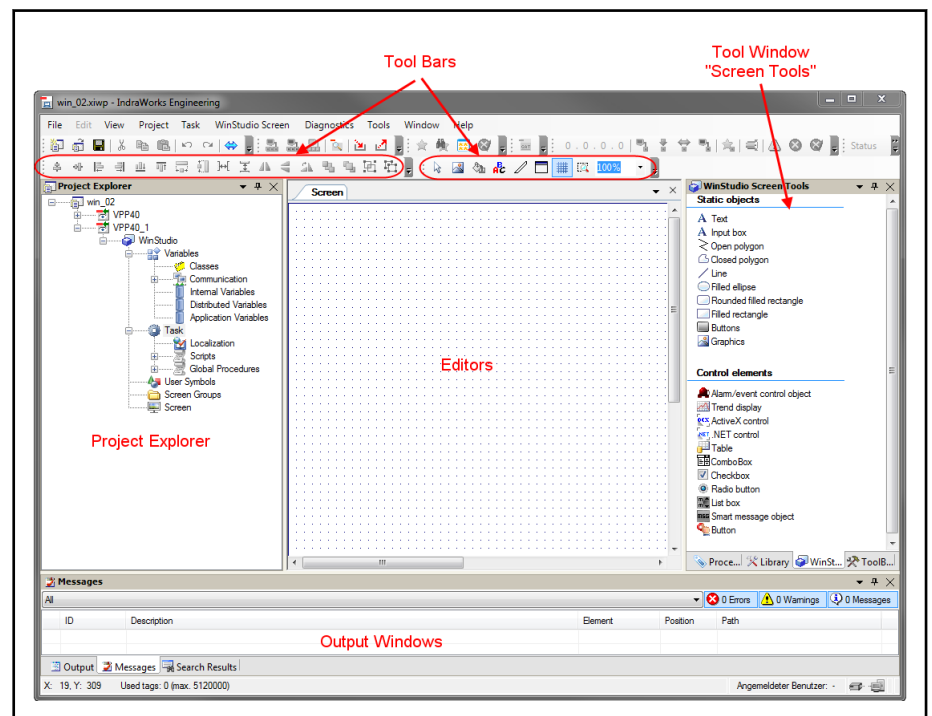


Fig.5-1: The WinStudio development environment in Indraworks

5.2 WinStudio Toolbars

5.2.1 WinStudio Toolbars, Customizing

The creation of WinStudio applications is supported by several toolbars with numerous context-related functions. An overview of the available toolbars is provided in the IndraWorks main menu under **Tools ► Customize**. A configuration dialog is displayed to activate toolbars that are used frequently or hide toolbars that are used less frequently.

Development Environment

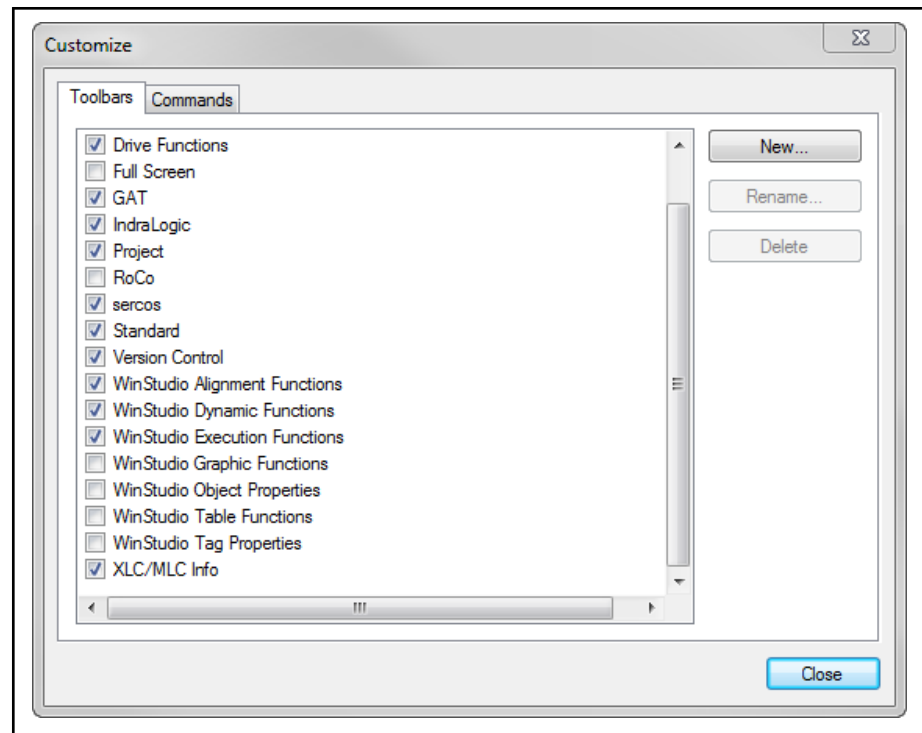


Fig.5-2: Available WinStudio toolbars



Added toolbars are displayed at a predefined position. Toolbars can be arranged at any time by clicking on the left edge of the toolbar and keeping the mouse button pressed to draw it to the desired position.

The WinStudio functions offered in the individual toolbars depend on the context.

They can only be activated if the respective editor is opened and one or several objects are selected.

All functions are additionally available with opened WinStudio Editor in the IndraWorks main menu under menu item "WinStudio <element name>", e.g. "WinStudio Math Sheet" or "WinStudio Screen". This way, it is not necessary to show toolbars that are not frequently used.

In the following, please find a description of the individual toolbars.

5.2.2 WinStudio Execution Functions Toolbar

With the "WinStudio Execution Functions" toolbar, basic functions of WinStudio screens opened in the editor can be tested.



Fig.5-3: WinStudio dynamic functions toolbar

Testing the screen

Activate the test mode of the currently edited WinStudio screen with the  **Test screen** button.

In test mode, animations (e.g. dynamic positioning or resizing of objects) can be tested.

Screen objects can be repositioned and properties changed.

However, no tasks are started. Thus, no background scripts and functions can be tested in this test mode that require communication with external devices.

Stopping the test

With the , **Stop test** button, the test mode is deactivated and the normal editing mode activated.

5.2.3 WinStudio Alignment Functions Toolbar

The "WinStudio Alignment Functions" toolbar offers numerous functions to align, group and design screen objects.



Fig. 5-4: "Alignment Functions" toolbar



Some functions in the toolbar are only available if one or multiple objects have been selected in the WinStudio screen.

Move the cursor over an active "tool" to display the function in plain text.

- **Center vertically** : Aligns all selected objects at the vertical center line of the object selected last
- **Center horizontally** : Aligns all selected objects at the horizontal center line of the object selected last
- **Left-aligned** : Aligns all selected objects at the left edge of the object selected last
- **Right-aligned** : Aligns all selected objects at the right edge of the object selected last
- **Align at bottom** : Aligns all selected objects at the bottom edge of the object selected last
- **Align at top** : Aligns all selected objects at the top edge of the object selected last
- **Same width for selected objects** : Increases/decreases the width of the object selected first to the width of the second one
- **Same height for selected objects** : Increases/decreases the height of the object selected first to the height of the second one
- **Evenly distribute horizontally** : Defines an equal horizontal spacing between a series of selected objects
- **Evenly distribute vertically** : Defines an equal vertical spacing between a series of selected objects
- **Mirror horizontally** : Rotates the selected object around an imaginary horizontal center line until it is a mirror image of the original object
- **Mirror vertically** : Rotates the selected object around an imaginary vertical center line until it is a mirror image of the original object
- **Rotate** : Rotates the selected object clockwise by 90 degrees

Development Environment

- **Move object to the background** : Moves one or several selected objects behind all other objects in the screen
- **Move object to the foreground** : Moves one or several selected objects in front of all other objects in the screen
- **Group objects** : Combines several selected objects to a single object
- **Ungroup** : Divides a grouped object into its individual components

5.2.4 WinStudio Tag Properties Toolbar



Fig. 5-5: WinStudio tag properties toolbar

Configure properties of WinStudio objects with the "WinStudio Tag Properties" toolbar.

- **Replace variable globally** : Enables a global exchange of a variable by an equivalent one
- **Change object properties** : Opens the object properties window of the selected object/object group

5.2.5 WinStudio Object Properties Toolbar

The "WinStudio Object Properties" toolbar offers functions to configure object properties.



Fig. 5-6: "Object Properties" toolbar



Some functions in the toolbar are only available if one or multiple objects have been selected in the WinStudio screen.

Move the cursor over an active "tool" to display the function in plain text.

- **Select object** :
- **Edit background screen** :
- **Fill color** :
- **Select font and text color** : Opens the selection window for the highlighted text/texts, in which fonts, style, text size, color and effects can be specified
- **Line color** : Opens the selection window for line color of highlighted objects and defines the default color for new objects
- **Background color** : Opens the selection window for the background color of the WinStudio screen

- **Activate/deactivate grid** : Controls whether the grid in the screen editor is displayed or hidden
- **Select zoom range**  100% : Controls the zoom process for the section highlighted with the magnifier / for the entire Winstudio screen

5.2.6 WinStudio Table Functions Toolbar



Fig.5-7: WinStudio Table Functions toolbar

The "WinStudio Table Functions" toolbar offers the following buttons:

- **Create trend objects** : Creates a trend object in the editor window
- **Create trend display** : Creates a trend display (alternatively to the control object with the same name) in the editor window
- **Change security level** : Opens the security dialog for a WinStudio screen. Also refer to [Security Access Levels on page 358](#)
- **Find text** : Enables searching of texts in the active window
- **Replace** : Opens the dialog window to replace "Only variables", "variables and properties", "user properties". Also refer to [Renaming and Removing Tags on page 135](#)
- **Add line** : Enables inserting lines into a worksheet
- **Select object** : Opens the dialog window to enter an object ID whose object is found in the active editor window

5.2.7 WinStudio Debug Functions Toolbar

The "WinStudio Debug Functions" toolbar supports testing and debugging of VB scripts. Scripts can be "debugged", i.e., execution can be stopped at a certain point, scripts can be executed in individual steps and values of variables can be observed.



Fig.5-8: WinStudio Debug Functions toolbar

- **Start debug mode** : Starts the WinStudio application in debug mode
- **Activate/deactivate breakpoints in the Viewer Task** : Activate this option to debug scripts that are executed in the Viewer Task These include:
 - Graphics Script
 - Screen scripts
 - Command scripts (e.g. on a button)
- **Activate/deactivate breakpoints in the Background Task** : Activate this option to debug scripts that are executed in the Background Task. These include:
 - Start script
 - Background scripts

Development Environment

- **Set/remove breakpoint** : Inserts a breakpoint into the currently opened script at the current cursor position or removes it



Breakpoints cannot be inserted at every position of scripts (e.g. blank lines, comment lines or lines with variable declarations). If the cursor is positioned in one of these lines and a breakpoint is set, it is automatically moved to the next admissible line.

- **Interrupt execution** : Interrupts the execution and switches to single step mode
- **Continue execution** : Leaves the single step mode and continues execution
- **Stepping** : Changes into the subfunction if the cursor is in single step mode on a line with function call
- **Next command** : Executes the next command
- **Return** : Returns from a subfunction to the superordinate function call
- **Stop debug mode** : Stops the debug mode

5.2.8 WinStudio Dynamic Functions Toolbar

With the "WinStudio Dynamic Functions" toolbar, properties of screen objects can be changed dynamically, i.e. animated.

To do so, select a screen object by clicking it and press one or several buttons of the toolbar. Press again to remove the dynamic property.



Fig. 5-9: WinStudio dynamic functions toolbar

- **Command** : Adds interfaces for commands that are to be executed by clicking on the screen object
- **Hyperlink** : Adds the "Hyperlink" property. Afterwards, a link (URL) can be configured that is opened when clicking on the screen object
- **Bar graph** : Adds the "Bar graph" property for filled rectangles, ellipses and polygons
- **Text input/output** : Inserts the "Text input/output" property to dynamically change the text displayed in a text object
- **Color change** : Adds the "Color change" property to change the background color according to a variable value
- **Position** : Adds the "Position" property to change the position of the screen object based on a variable value

- **Resizing** : Adds the "Resizing" property to change the size of the screen object according to a variable value
- **Line rotation** : Adds the "Line rotation" property to lines and polygons to rotate the object according to a variable value.

5.3 Project Explorer

5.3.1 Project Explorer, General Information

The "Project Explorer" organizes all of the screens, worksheets and other elements that make up the project and displays them in tree structure.

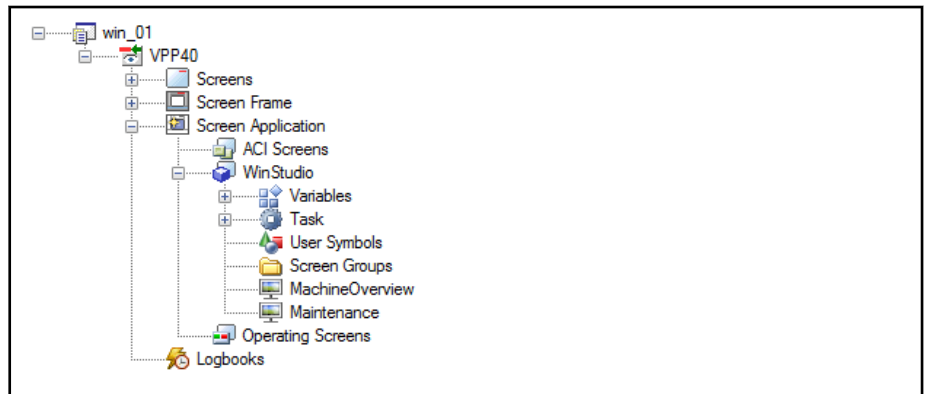


Fig. 5-10: The WinStudio folder structure

Click on the expand ► icon or double-click the folder to view the contents of the folder.

Click on the reduce ◀ icon or double-click the folder to close it.

All WinStudio elements in the project tree have a context menu. This menu can be opened by right-clicking on the element. The context menu contains general functions as well as functions that are only available for the respective element type.

The individual sections of the WinStudio project tree include different WinStudio functionalities.

5.3.2 "WinStudio" Folder

The folder **WinStudio** includes all screens, screen groups and symbols in the project.

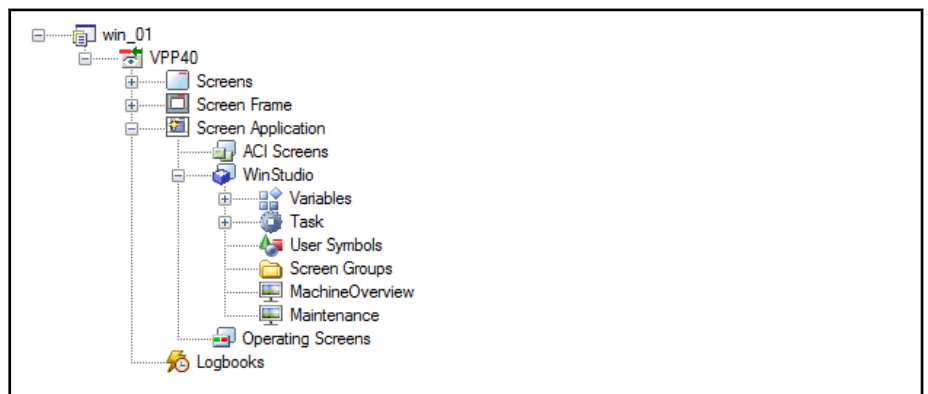


Fig. 5-11: The "WinStudio" folder

The folders on the "Graphics" tab are described on the following pages:

Development Environment

- **WinStudio** contains all [screens on page 156](#) created for the current project
- **Variables** contains all functions and editors for editing variable types as well as configuration of process integration of variables into external devices
- **Tasks** contains different editors and functions required to create more complex applications ("math sheets", "alarms", etc.)
- **Screen Groups** contains all [screen groups on page 161](#) (individual WinStudio screens combined into manageable groups) created for the current project
- **User Symbols** contains all [user-defined symbols on page 200](#), which can be groups of screens and/or text. Customized symbols for the project can be created and saved to this folder
- **Graphics Script** contains [predefined functions that are executed when certain screen actions occur on page 718](#), such as when the Thin Client is started on a remote station

5.3.3 "Variables" Folder

The **Variables** folder contains all functions and editors required to manage WinStudio tags, to create user-defined data structures ("Classes") as well as for communication connection to a control system by means of one of the different available drivers.

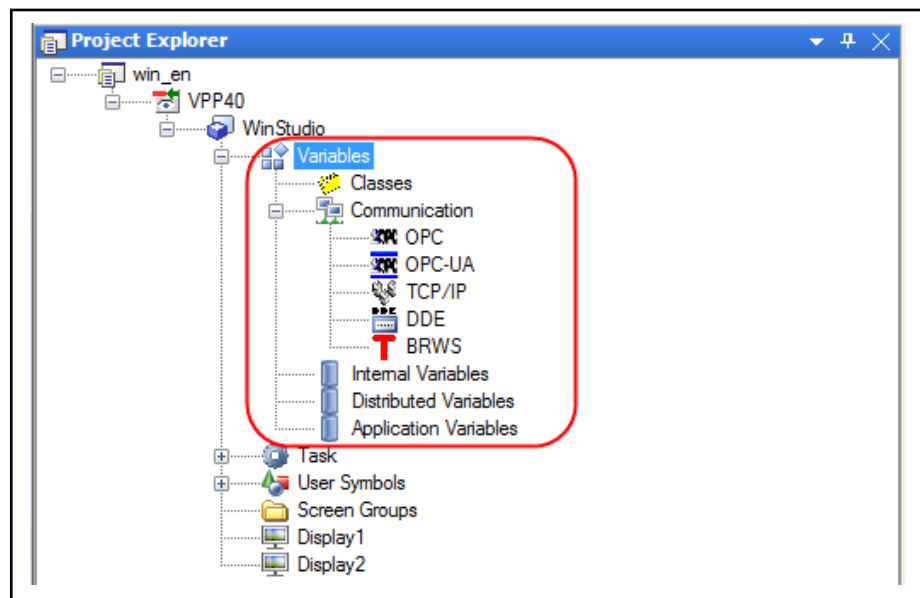


Fig. 5-12: "Variables" folder

This folder contains the following elements:

- [Classes on page 108](#) contains compound tags, referred to as "classes" tags, created to link a set of values (rather than a single value) with an object
- [System Tags on page 110](#) contains predefined tags with particular functions that are used by the project for specific monitoring tasks (for example, "Date" tags contain the current date in string format). All internal tags are "read-only", which means they cannot be added, edited or removed from the database
- [Shared Database on page 109](#) contains tags automatically created for process integration

- [Project Tags on page 106](#) contains tags created during project development (such as screen tags or tags that read from/write to connected devices)
- [Procedures on page 716](#) contains VBScript functions and subroutines that can be called by any other script in the project
- [Event Logger on page 258](#) contains features to log or read out events
- [Translation on page 381](#) contains the translation worksheet that defines how the project's user interface is to be translated into other languages

5.3.4 "Tasks" Folder

The **Tasks** folder organizes the worksheets that are executed as Background Tasks during project runtime.

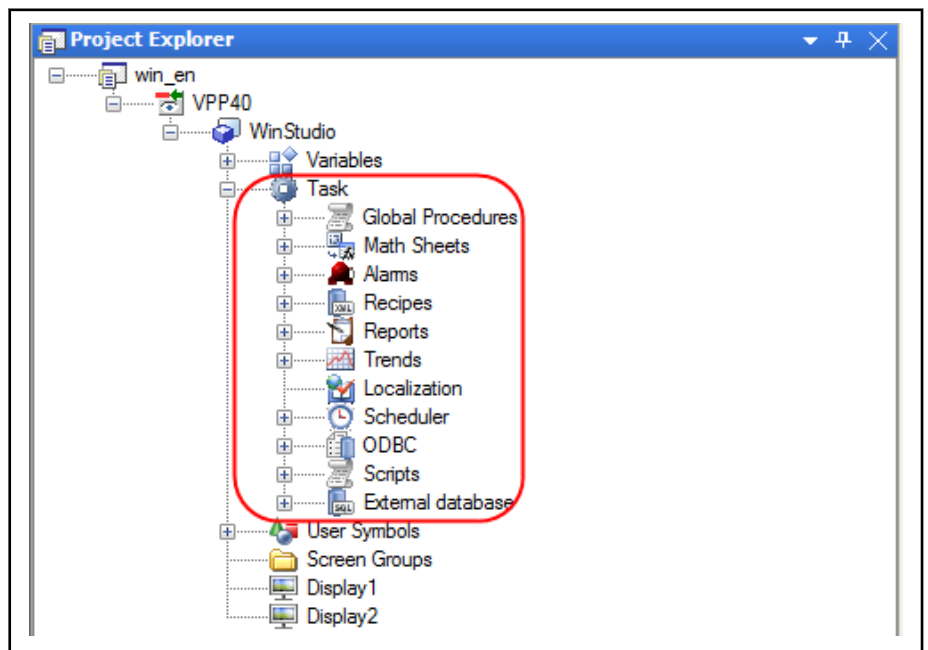


Fig.5-13: "Tasks" folder

The folders on the Tasks tab are described on the following pages:

- **Math Sheets** contains the [math worksheets on page 324](#) used to configure and create additional routines that work with different tasks. WinStudio executes "math" worksheets as Background Tasks during runtime. Configure "math" worksheets to provide sequences and mathematical calculations for the project
- **Alarms** contains [Alarm worksheets on page 247](#) to specify alarm messages that are displayed during runtime if an assigned tag reaches or exceeds a certain value. The pending alarm messages at runtime can be logged automatically into a file and displayed in a WinStudio screen by means of the alarm control
- **Recipes** contains [Recipe worksheets on page 319](#) to configure the variables the data of which is written to an ASCII or XML file or read from a file and applied to the project database at runtime
- **Reports** contains [Report worksheets on page 320](#) to create text documents that are created at runtime and sent to a printer or file. Not only fixed texts but also current values of variables can be put out in reports. This function is particularly useful for documentation of system data
- **Trends** contains [Trend worksheets on page 269](#). Trend worksheets enable recording of time characteristics of variable values. These se-

Development Environment

quences of values can be used to display them in a WinStudio screen with Trend Control. Declare those variables in Trend worksheets that are to be saved to what is referred to as a binary history file (*.hst)

- **Localization** opens the translation editor. Here, all texts that are used in the application can be translated to different languages. At runtime, the texts can be switched between various languages
- **Scheduler** contains [Scheduler worksheets on page 326](#) to execute a function on various events. Events can be value changes of variables, reaching a value or defined times or dates
- **ODBC** contains [ODBC worksheets on page 321](#) to configure ODBC interfaces in a network environment. ODBC worksheets are used to exchange data between projects and any database that supports the ODBC interface
- **Scripts** contains the [Startup script on page 325](#) as well as the **Graphics script**. [Script groups on page 324](#). These predefined scripts can be used to execute functions on start of the application or during execution of the Viewer. Additionally, user-defined scripts can be created in this folder to program sequences in VB Script
- **Global Procedures** contains the **Main Procedures** element by default. In this file, user-defined functions can be declared that can be called in a script. To structure these functions, add further scripts to declare additional functions
- **External database** contains [Database worksheets on page 328](#) that communicate with external databases via the standard ADO.NET interface (as an alternative to ODBC)

5.3.5 Communication Folder

Create worksheets in the subfolders of the **Communication** folder to establish the communication with other devices or software by means of available protocols.

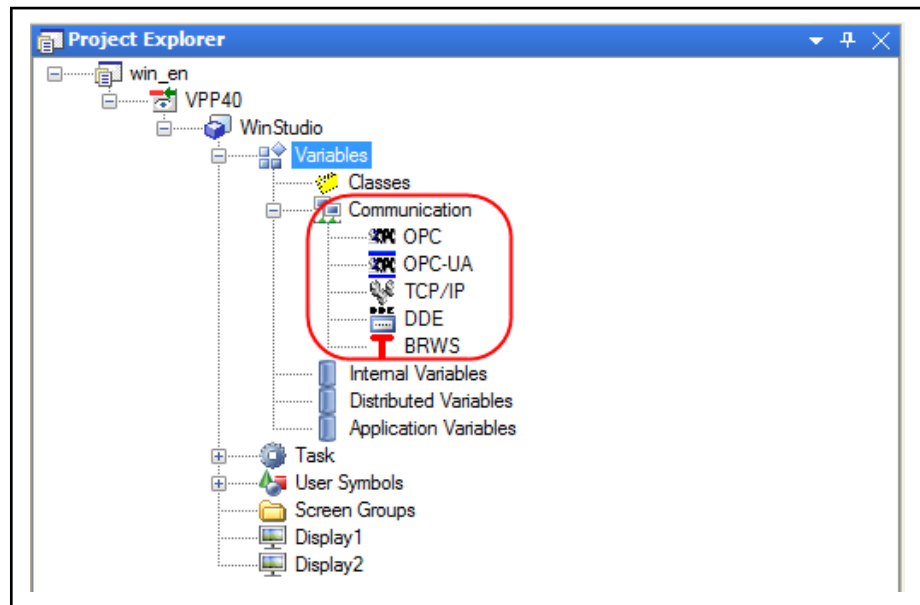


Fig.5-14: "Communication" folder

The Communication folder contains the following subfolders and functions.

- **OPC** contains [OPC worksheets on page 342](#) to configure communications between a project (which acts as an OPC Client) and any other device that implements the classic OPC Standard (OPC-DA)
- **OPC UA** contains [OPC UA worksheets on page 345](#) that are used to establish a connection to OPC Servers via the new "OPC Unified Architecture" protocol
- **TCP/IP** contains [TCP/IP worksheets on page 351](#) to configure TCP/IP communication to other "WinStudio" stations

By means of "WinStudio" TCP/IP client and server modules, databases of two or more projects can be synchronized

- **DDE** contains [DDE worksheets on page 352](#) to enable WinStudio to exchange data via DDE protocol (Dynamic Data Exchange) with a Windows application (e.g. Microsoft Excel or similar Windows program supporting this interface). DDE communication is an interaction between server and client programs. "WinStudio" provides interfaces that run as clients or as servers
- **BR_WS** contains [Driver worksheets on page 333](#) to enable WinStudio to exchange data with a PLC control via the BR_WS communication interface. A "communication driver" is a .DLL file that contains specific information about the remote equipment and implements the communication protocol
- **Additional drivers:** If data is to be exchanged with a device that does not support any of the protocols described above, an additional driver can be added. If a driver file (.DLL) is available for the device, copy it to the ..\WinStudio\Drv folder. Subsequently, open the driver configuration dialog in the context menu of the "Communication" folder via the menu item **Add/Remove Driver**. All available drivers are listed under this item. After adding a driver, the respective subfolder is created in which the associated worksheets can be created

5.4 Configuring and Testing an Application

5.4.1 Application Verification

Select **Verify Application** to recompile all [math worksheets on page 324](#), expressions, and screen logics, as well as to update HTML files using the settings configured in the [Web tab of the Project Settings dialog on page 78](#).

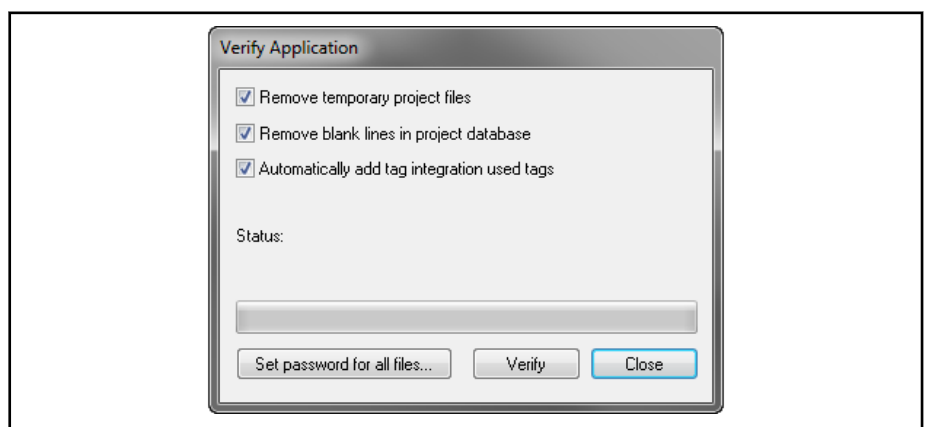


Fig. 5-15: Verifying an application

Use this dialog to reassign the password for all files:

Development Environment

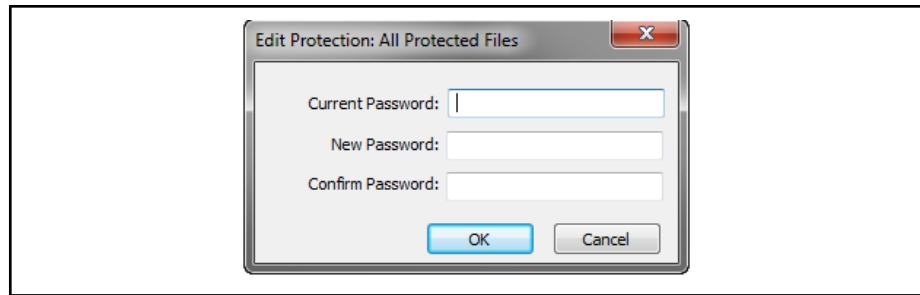


Fig. 5-16:

If any variables are used in the application that are not defined in the database, an error message indicates where (i.e. the screen or worksheet) those tags are used.



When saving screens or worksheets, WinStudio adds a pointer to the current database version. When executing a project, WinStudio compares the database version with the current project database and recompiles the expressions if they do not correspond to the database.

To avoid this compilation during the project runtime, it is recommended to execute the **Verify Application** tool before downloading/finishing a project.

This function is also to be used when converting a project to a new WinStudio version.

In order to save disk space and clean the project before downloading it to the runtime station select **Verify Application** to delete all temporary files (e.g., *.txt, *.mac and *.tag files) from the project folder.

However, WinStudio does "not" automatically upload the project to the runtime station. To transfer the project to the runtime station, use the **Send project to target** tool.

5.4.2 Starting the Application

In the context menu of the "WinStudio" folder, click on **Start application** to execute the runtime modules that are marked as **Automatic** in the <Execution Tasks> dialog ("Execution Tasks" tab).

Start the "Viewer" module to open the screen(s) currently being edited.

If no "automatic" tasks are specified, "WinStudio" automatically starts the "Viewer" and "Background" tasks when clicking on **Start application**.

If no screens are opened in the editor, the "Viewer" module opens the configured start screen (refer to the **Start screen** field on the "Viewer" tab in the [Project Settings on page 62](#) dialog).

Start application refers to the project on the target system (configured in the [Remote Management on page 61](#) dialog). Before clicking **Start application**, make sure that the correct target system is configured (local or remote).

5.4.3 Stopping the Application

In the context menu of the "WinStudio" folder, click on **Stop application** to stop all runtime tasks.

Stop application refers to the project of the selected target system (refer to the [Remote Management on page 61](#) dialog). Before clicking **Stop application**, make sure that the correct target system is configured (local or remote).

5.4.4 WinStudio Target Platform

Select this function to open the "Remote Management" dialog to manage remote stations (download/upload files and run/stop remote projects).

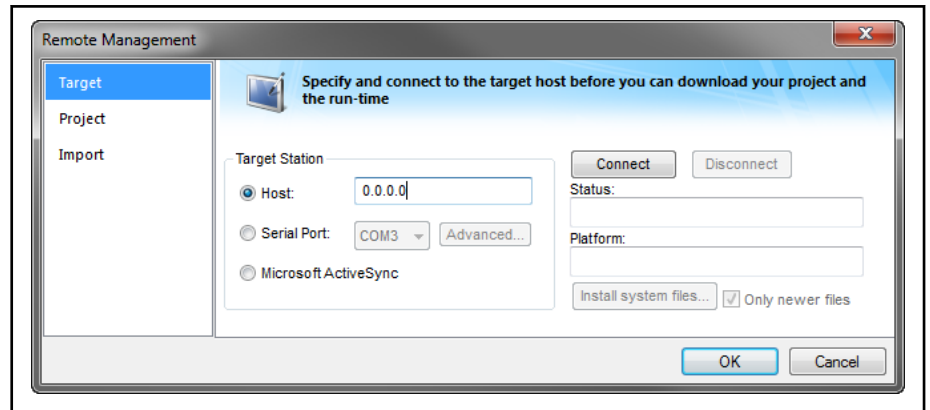


Fig.5-17: Remote Management dialog

This dialog contains the following tabs:

- **Target:** Use this tab to specify the target system, including:
 - **Network IP, Serial Port, or Microsoft ActiveSync** (Microsoft ActiveSync currently only works with Windows Embedded Compact and Windows Mobile 5.0 and later.)
 - **Connect/ Disconnect** the target system
 - Review station **Status**
 - Returns the **Platform** (operating system + processor type) of the target system after connecting to it
 - **Install system files** (or **Only newer files**)
- **Project:** Use this tab to specify the project, including:
 - **Local and Target** directory paths
 - Specify whether to **Send** (the project) **to target system**
 - **Only newer files:** If this option is enabled, only newer application files are sent to the device
 - **Retain user files:** If this option is enabled, any user files in the project folder are retained
 - **Send file**
 - **Run/Stop** the project
 - Review project **status**
- **Import:** Use this tab to import a project, this includes:
 - Specify the **target** and **source** project path
 - **Get from target**
 - **Get log file**
 - Review project **status**

For further information on connecting to Windows Embedded devices, refer to [Installing CEView on a Windows Embedded device on page 39](#).

Development Environment

5.4.5 Project Settings

Information Tab

Select the entry **Properties** in the context menu of the "WinStudio" folder to open the "Project Settings" dialog.

It controls settings concerning the overall project.

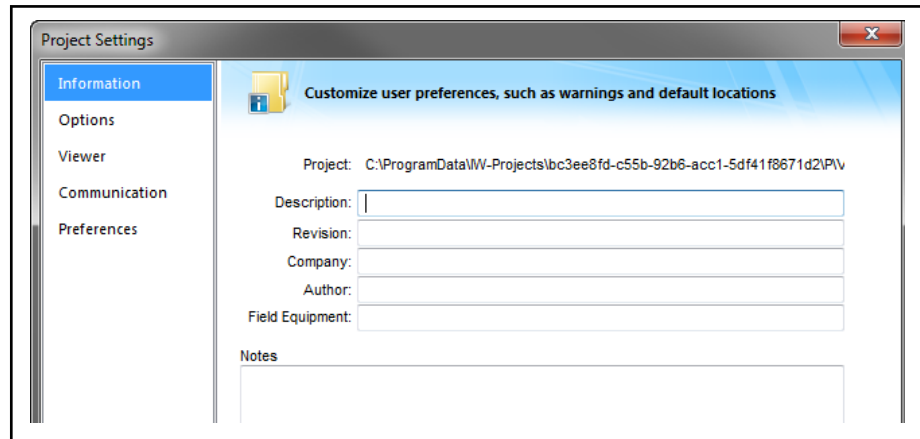


Fig.5-18: Project Settings: Information tab

Identify the project with the following parameters (for documentation purposes only).

- Description
- Revision
- Company
- Author
- Field Equipment
- Notes

Options Tab

Options Tab, General Information

Use this tab to specify parameters relating to the project in general.

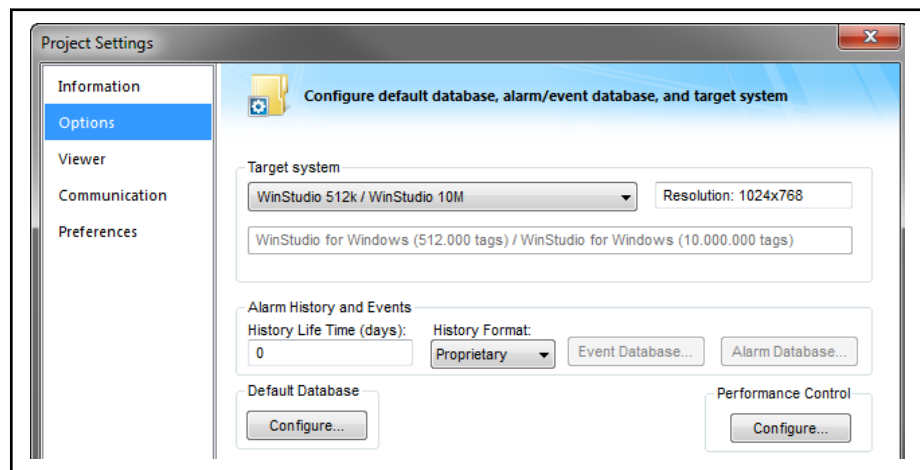


Fig.5-19: Project Settings: Options tab

These parameters are described in the following:

- **Target system:** Use the selection list to specify the target system for the current project. The target system sets the project restrictions (such as

Development Environment

number of tags supported) and has to match the active license. The description of the main license restrictions for each target system is displayed below the selection list where it is specified.

NOTICE

If a Target system level is specified that does not match the current license level on the target system, the project may not run properly.

- **Resolution:** Displays the screen resolution of the project.
- **Alarm History and Events:** Enter a value into the "History Life Time (days)" field to specify how long alarm and event history files are retained. After the specified number of days, the project automatically deletes existing alarm/event history files that are older than the period specified. Enter zero into the field to configure that the project does not delete any history files automatically. In this case, create an external procedure to delete the old history files. Otherwise, the free disk space in the computer decreases.
- **History Format:** Select the format of the Alarm/Event history as follows:

Format	Description
Proprietary	Saves the history data in the "Alarm" subfolder of the project folder (by default) in text files using the proprietary format
Database	Saves the history data in the user-defined SQL relational database using the built-in ADO interface
Binary	Saves the history data in the "Alarm" subfolder of the project folder (by default) in binary files using the proprietary format

For more information, refer to [Saving the alarm history/event log to an external database on page 254](#).

- **Default Database:** Use this item to configure a default database that can be used by different tasks and objects. For further information, refer to [Configuring a Default Database for All Task Histories on page 442](#).

The PC-based control has a customized interface that requires information on the PC-based control application in order to share tags with the "WinStudio" project.

Saving the Alarm History to an External Database

By default, the alarm history and event log are saved to proprietary-format text files in the Alarms folder of the project. However, the project settings can be modified to save them to an external SQL database instead.

1. Open the **Options** tab in the "Project Settings" dialog.

Development Environment

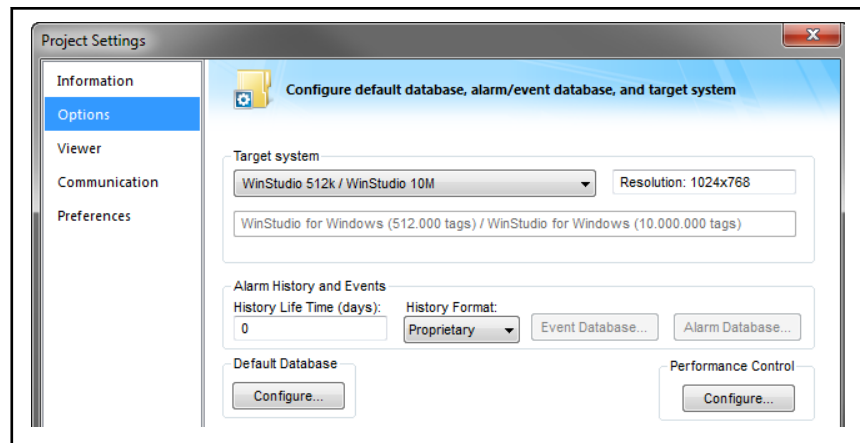


Fig.5-20: Project Settings: Options

2. In the **Alarm History and Events** pane, in the **History Life Time (days)** input field, enter the number of days of history that are to be saved. As the history exceeds the specified number of days, it is automatically deleted in a first-in, first-out manner. If no number is specified - that is, if it is left blank or set to 0, - the history is never deleted. There is no limit on how much history can be saved, however, the more is saved, the more disk space it takes.
3. From the **History Format** list, select **Database**.
4. To configure a single, default database to be used for both the alarm history and the event log (as well as all other runtime tasks), click **Configure** in the **Default Database** pane. The "Default Database Configuration" dialog is displayed. Configure the database connection using this dialog. For further information, refer to [Configuring a Default Database for All Task Histories on page 442](#).
5. To configure a separate database, either for the event log or the alarm history, click **Event Database** or **Alarm Database**, respectively. In either case, a "Default Database Configuration" dialog is displayed. Configure the database connection using this dialog. For more information, refer to [Default Database Configuration on page 438](#).
6. Click on **OK**.

Configuring a Default Database for All Task Histories

A default database can be configured to save the historical data of all tasks in a project. After configuring a default database, either use the default database or configure a new database for this special worksheet when creating a new task worksheet.

Configuration of the connection settings for the default database:

1. Open the properties dialog of the "WinStudio" folder and select the "Options" tab
2. In the "Default Database" pane, click **Configure**. The "Default Database Configuration" dialog is displayed

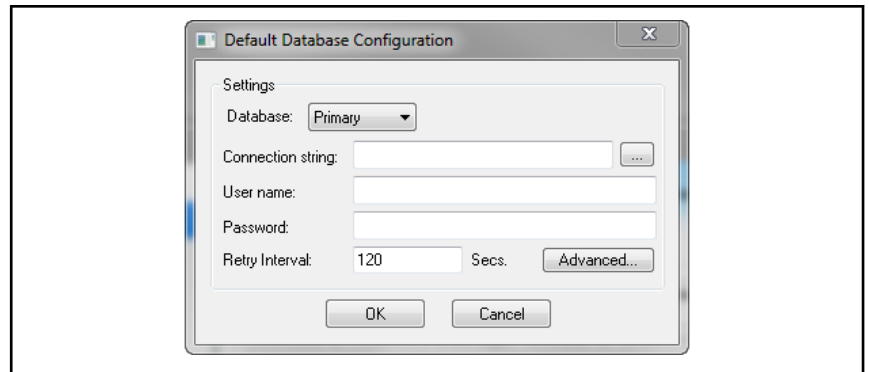


Fig.5-21: Default Database Configuration dialog

Please refer to [Default Database Configuration dialog on page 438](#) for help regarding the completion the fields in this window.

Database Configuration

Database connection configuration

The data exchange with an external data base file is carried out analogously to the device drivers. First, configure a new data base connection. Afterward, worksheets can be created in which the created data base connection can be selected and WinStudio tags can be linked to individual data sets of the external database.

To configure a new data base connection, open the context menu of the folder **WinStudio ► Tasks ► External database** and select the entry "Default Database Configuration".



The "External database" folder is only available if "WinStudio 1.5k" or higher is selected as target system in the WinStudio properties and if the device version in the VPx/Vex properties is "13V06" or higher.

Open the configuration dialog for a new database connection using the "Add" button in the overview dialog. Enter a name in the "Name" field that can later be selected in the worksheets.

The "Database Configuration" dialog enables configuration of the necessary settings to link "WinStudio" to an external database file.

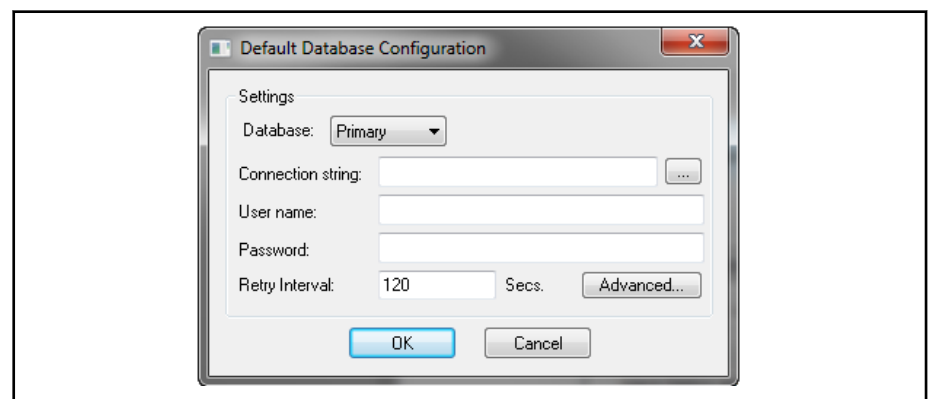


Fig.5-22: Default Database Configuration dialog

- **Database** combo box: To select either Primary or Secondary. If "Primary" is selected, all settings of the Default Database Configuration apply to the Primary Database interface. Otherwise, they apply to the Secondary Database interface. The Secondary Database can be configured in the following modes:

Development Environment

- **Disabled:** In this mode, "WinStudio" saves data in the Primary Database only. If the Primary Database is unavailable for any reason, the data is not saved anywhere else. This option may cause loss of data if the Primary Database is not available.
- **Redundant:** In this mode, "WinStudio" saves data in both Primary and Secondary Databases. If one of these databases is unavailable, "WinStudio" keeps saving data only in the database that is available. If the database that was unavailable becomes available again, "WinStudio" synchronizes both databases automatically.
- **Store and Forward:** In this mode, "WinStudio" saves data in the Primary Database only. If the Primary Database becomes unavailable, "WinStudio" saves the data in the Secondary Database. If the Primary Database becomes available again, "WinStudio" moves the data from the Secondary Database into the Primary Database.



The Primary and Secondary databases can be different types of databases. However, they have to have the same fields.

Use the Secondary Database to increase the reliability of the system and to use the Secondary Database as a backup when the Primary Database is not available. This architecture is particularly useful when the Primary Database is located in the remote station. In this case, a Secondary Database can be configured in the local station in such a way that data is saved temporarily if the Primary Database is not available (during a network failure, for instance).

- **Use project default:** If this option is checked, "WinStudio" uses the settings in the Default Database for the configured task (Connection string, User name, Password, Retry Interval and Advanced). If this option is disabled, these settings can be configured individually for the current task.
- **Connection string:** This field defines the database where "WinStudio" writes and reads out values as well as the main parameters used when connecting to the database. Instead of entering the connection string manually, the browse button (...) can be clicked to select the database type from the **Data Link Properties** window.

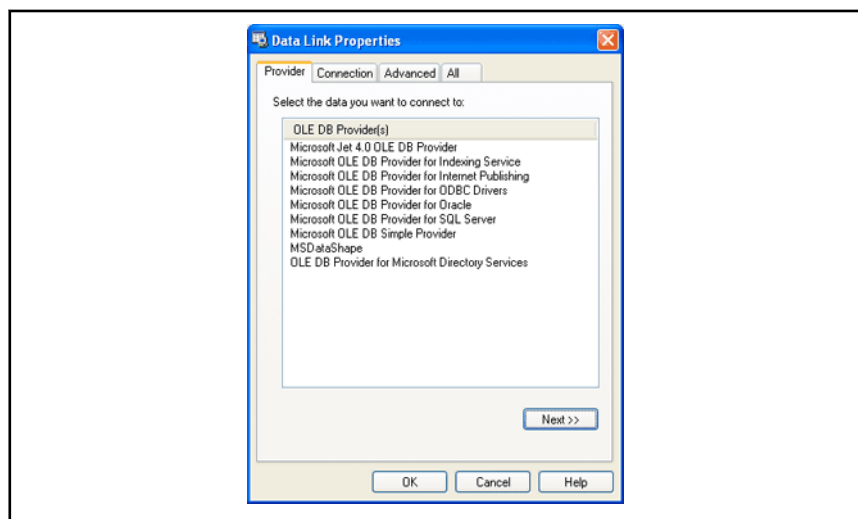


Fig.5-23: Data Link Properties dialog



The list of database providers shown in the Data Link Properties window depends on the providers actually installed and available on the computer where "WinStudio" is running. Consult the operating system documentation (or the database documentation) for further information regarding the settings of the database provider for the used database.

- **User name:** User name used to connect to the database. The user name entered in this field has to match the user name configured in the database.
- **Password:** Password used to connect to the database. The password configured in this field has to match the password configured in the database.
- **Retry Interval:** If "WinStudio" is unable to connect to the database for any reason, it automatically retries to connect to the database after the seconds configured in this field have passed.
- **Advanced:** Pressing this button provides access to the configuration of advanced settings. For most projects, the default value of these settings does not need to be modified and is to be retained.

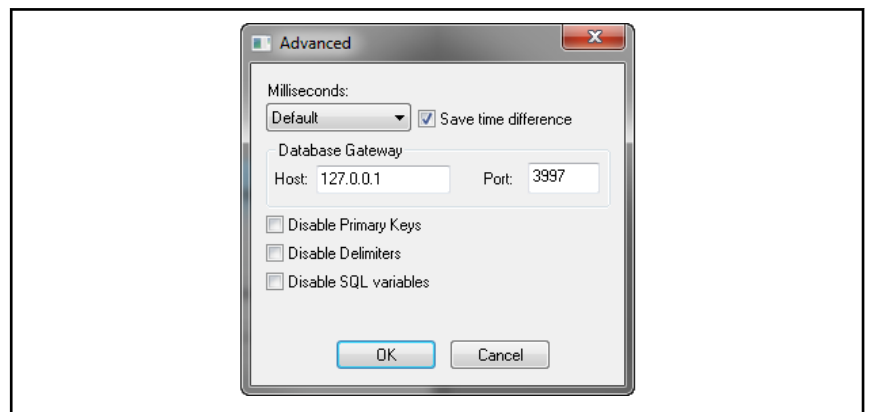


Fig.5-24: Database configuration: Advanced dialog

– Milliseconds

Use this item to set how the milliseconds are saved when saving the date in the database. Different databases save the date in different formats. For example, some databases do not support milliseconds in the **date field**. The following options are available:

- **Default:** Uses the pre-defined format for the current database. The databases that were already tested by "Bosch Rexroth" are preconfigured with the most suitable option. When selecting "Default", "WinStudio" uses the preconfigured setting for the current database type. If a database is used that has not been preconfigured, it is tried to save the milliseconds in a separate field if the Default option is selected.



The default option for each database is pre-configured in the **STADOSvr.ini** file, stored in the **\BIN** sub-directory of "WinStudio". For further information on configuration of the **STADOSvr.ini** file, refer to [Studio Database Gateway on page 432](#).

Development Environment

- **Disable:** Does not save the milliseconds at all when saving the date in the database.
- **Enable:** Saves the milliseconds in the same field where the date is saved.
- **Separate Column:** Saves the milliseconds in a separate column. In this case, the date is saved in one field (without milliseconds) and the number of milliseconds is saved in a different column. This option is displayed if time stamps are to be saved with milliseconds precision and the used database does not support milliseconds for **date fields**.
- **Save time difference:** When this option is activated (default), "WinStudio" saves the time zone of the computer where the project is running in each register of the database. This option has to be enabled to avoid problems with daylight savings time.
- **Database Gateway:** Enter the host name/IP address of the network node where the "WinStudio" Database Gateway is running. The TCP port number can also be specified, however, if the default is not used, the "WinStudio" Database Gateway has to be configured with the same TCP port. For further information on configuration of the advanced settings of the "WinStudio" ADO Gateway, refer to [Studio Database Gateway on page 432](#).
- **Disable Primary Keys:** For some modules, "WinStudio" tries to define a primary key to the table in order to speed up the queries. Enable this option if a database is used that does not support primary keys (e.g. Microsoft Excel).
- **Disable Delimiters:** Select this error handling option to disable the delimiters that are used to format communications with the database. Delimiters can cause problems when a Trend Control or grid creates a query that includes aggregates such as Min and Max.
- **Disable SQL variables:** Select this error handling option to disable SQL variables, such as "@Value1" and "?", that are often used in SQL commands and requests. Some specific database providers do not support these variables.

Table pane This pane enables configuration of the settings of the table where the data is saved. All tasks can share the same database. However, each task (Alarm, Events, Trend worksheets) has to be linked to its own table. "WinStudio" does not check this field for invalid configuration. For this reason, ensure that the respective setting is suitable for the used database.

- **Use default name:** If this option is checked (default), "WinStudio" saves and/or loads the data in the table with the default name written to the **Name** field.
- **Create automatically:** If this option is enabled (default), "WinStudio" automatically creates a table with the name displayed in the **Name** field. If this option is disabled, "WinStudio" does not automatically create the table. For this reason, data can only be saved in the database if the database already contains a table with the name that had been preconfigured manually in the field **Name**.
- **Name:** Specifies the name of the table from the database where the history data are saved. To specify a worksheet in a Microsoft Excel spreadsheet file, use the following syntax:
[worksheetname\$]

Development Environment

- **Refresh:** If the set database is currently available, press the **Refresh** button to fill the **Name** combo box with the name of the tables currently available in the database. This way, it is possible to select the table where the history data is to be saved instead of manually entering the table name in the **Name** field.

Runtime pane

In this pane, the runtime values can be defined. The following fields are available:

- **Status (output):** The tag in this field receives one of the following values:

Value	Description
0	Disconnected from the database. The database is not available, its configuration is incorrect or an incorrect operation is executed
1	The database is successfully connected
2	The database is being synchronized

- **Reload (output):** Specify a "reload" tag when using curly brackets in the configuration field. To reconnect to the database using the updated values of the tags, set the tag on this field to 1. "WinStudio" updates the configuration when trying to execute an action in the database, setting the tag back to 0 when finished.

For further information, refer to:

[Configuring a Default Database for All Task Histories on page 442.](#)

Viewer Tab

Viewer Tab, General Information

Use this tab to configure the project viewer and to change certain behaviors at runtime.

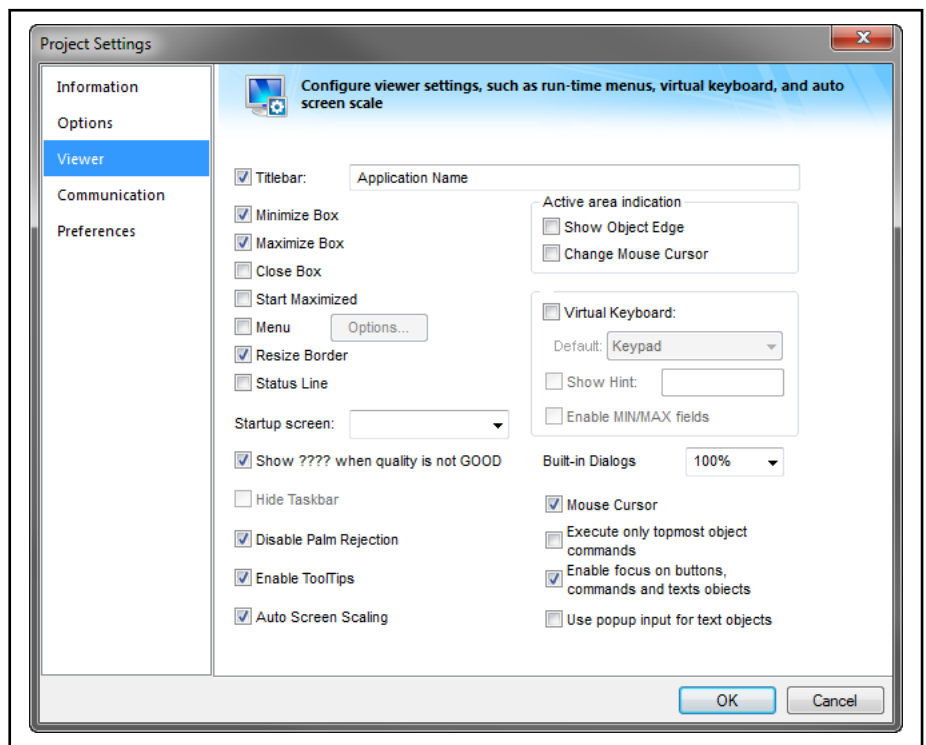


Fig. 5-25: Project Settings: Runtime Desktop tab

Development Environment

- **Title bar:** Click (enable) this box and enter a new name into the field provided to specify or change the default title bar text for the Viewer window
- **Minimize Box, Maximize Box, and Close Box:** Click these checkboxes to display (enable) or hide (disable) these buttons in the Viewer window
- **Start Maximized:** Click (enable) this checkbox to maximize the Viewer window automatically when running the project
- **Menu and Options...** button: Click (enable) the checkbox and the **Options...** button to specify which menu options are available at runtime. If the "Runtime menu options" dialog is displayed (as follows), click the checkboxes to display (enable) or hide (disable) the respective menu options.

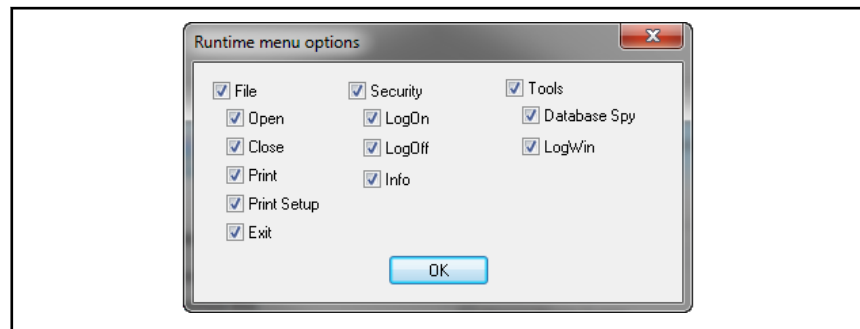


Fig.5-26: Runtime menu options dialog

- **Resize Border:** Click (enable) this checkbox to be able to resize the Viewer window at runtime
- **Status Line:** Click (enable) this checkbox to display the status line in the runtime project
- **Startup screen:** Click on the selection list and select a WinStudio screen or a screen group that is to be displayed automatically on opening the runtime project



Another way to specify a screen or screen group as startup screen is to right-click on the element in the "Project Explorer" element and to choose **Set as startup screen**.

- **Show ??? when quality is not GOOD:** Click (enable) this box to display question marks (???) instead of the tag value if the quality is not good
- **Hide Taskbar:** Click (enable) this box to hide the Windows taskbar by default
- **Disable Palm Rejection:** Select this option to disable Palm Rejection during project runtime. Palm Rejection is a feature of Windows 7 touchscreen devices that detects and rejects accidental touches from the operator's palm. However, the responsiveness of the touchscreen can be considerably reduced. For this reason, disabling this function can improve the performance of the project runtime object



Palm Rejection is only available on Windows 7 touchscreen devices, so selecting **Disable Palm Rejection** has no effect on projects running on other target systems.



Windows 7 touchscreen devices also have an option that makes it possible to press and hold on the screen to perform a right click ("press and hold"). This function is useful on final consumer devices like tablets and smartphones, however, it is possibly not useful when working on "WinStudio" projects as it is necessary to press and hold buttons and other screen objects. This option is enabled by default. To deactivate this option on the device, proceed as follows:

1. Click on the Start button and afterwards on Control Panel in the Start menu. The Control Panel window is displayed.
2. Double-click Stylus and Touch. The Stylus and Touch dialog is displayed.
3. Click the Touch tab.
4. From the Touch actions list, select "Press and hold" and click Settings. The Press and Hold Settings dialog is displayed.
5. Disable "Enable press and hold for right click".
6. Click OK to close the Press and Hold Settings dialog.
7. Click OK to close the "Stylus and Touch" dialog.

- **Enable ToolTips:** Click (enable) this checkbox to see Windows ToolTips when running the project. Tooltips can be configured in the **Hint** field of the "Object Properties" dialog of each object.
- **Auto Screen Scaling:** Click (enable) this box to automatically adjust WinStudio screens when changing the size of the Viewer window. This function is available for local projects running on Windows PC or in the Thin Client. "This parameter is not available for projects running on Windows Embedded target systems."
- "Active area indication" pane: Click (enable) the **Show Object Edge** and **Change Mouse Cursor** checkboxes in this area to modify the object edge and the mouse cursor when moving the cursor over any object where the command animation has been applied.
- **Virtual keyboard:** If this option is enabled, the [Virtual Keyboard on page 73](#) is enabled for the project. (This option is "not" available if the project is running on a Thin Client. For more information, refer to [Project Settings: Web on page 78](#).) With this keyboard it is possible to enter data during runtime on touchscreens without entering data on a physical keyboard. For example, a [text object on page 175](#) the [Text Data Link animation on page 225](#) was applied to and the **input enabled** option was enabled. A default configuration for the virtual keyboard can be set:
 - **Default:** Select the default keyboard type to be used in the project if no keyboard type is specified by the calling object or function.
 - **Scaling:** This option enables shrinking or enlarging the keyboard to fit the size of the target screen. A value of 100 % represents the default size of each keyboard type.
 - **Show Hint:** If this option is enabled, a hint is displayed in the title bar of the keyboard. For individual objects, the hint is configured in the Object Properties dialog. Otherwise, enter a string value into the "Show Hint" field to serve as a default hint.
 - **Enable MIN/MAX fields:** If this option is enabled, the permissible minimum and maximum values are displayed at the bottom of the keyboard. For objects, these values are configured in the Object

Development Environment

Properties dialog. Otherwise, the Min and Max properties of the associated tag are used by default.



The Min and Max fields are only displayed on the Keypad keyboard type and only if the associated tag is defined as Integer or Real. If the Min value exceeds the Max value, the input is disabled. If the Min/Max values configured on the object differ from the Min/Max values configured in the tag properties, the project attempts to convert the input accordingly.

- **Mouse Cursor:** Click (enable) this checkbox to display the mouse cursor in the runtime project.



The Mouse Cursor option is not supported in Windows CE running on Armv4l processors.

- **Execute only topmost object commands:** This option controls the project behavior if it is clicked in a pane where two or more screen objects overlap. If this option is enabled, only the commands of the topmost object are executed. If this option is disabled, the commands of all overlapping objects are executed.



The topmost object is the one with the highest ID number. (The ID number of an object is displayed in the [Status bar on page 87](#) at the bottom of the development environment.) The [Move to Back / Move to Front on page 234](#) and [Move Backward / Move Forward on page 235](#) tools can be used to change the order in which objects are stacked.

- **Enable focus on buttons, commands and text objects:** This option is enabled by default. If this option is enabled, clicking on a command button or text input during runtime puts the focus on that object, as shown in the illustration below:

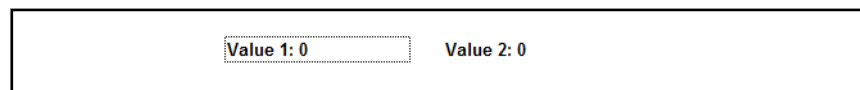


Fig.5-27: Focus on text input (left), no focus on text input (right)

After that, the **Tab** key can be used to navigate to all focusable objects in the screen or the **Enter** key can be used to activate the currently focused object (i.e., clicking on a command button, entering text in a text field), similar to a standard Windows application. This is useful if the Client workstation is equipped with a physical keyboard and if many objects are to be processed quickly.

To force the end-user to always use the mouse or touchscreen to activate screen objects, delete this option.

- **Use popup input for text objects:** Select this option to display a small popup for text inputs, as shown in the following illustration:

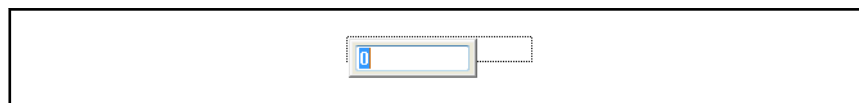


Fig.5-28: Popup for text input

Development Environment

This is an alternative to typing directly into the text object and to displaying a virtual keyboard for input (which requires using a mouse or touchscreen).



If the **Virtual Keyboard** option (above) is selected it overrides this option.

Moreover, if the **Enable focus on buttons, commands and text objects** option (above) is "disabled", this option is automatically enabled. This is to ensure that there is an indication in the screen of which text input is currently activated.

Keyboard

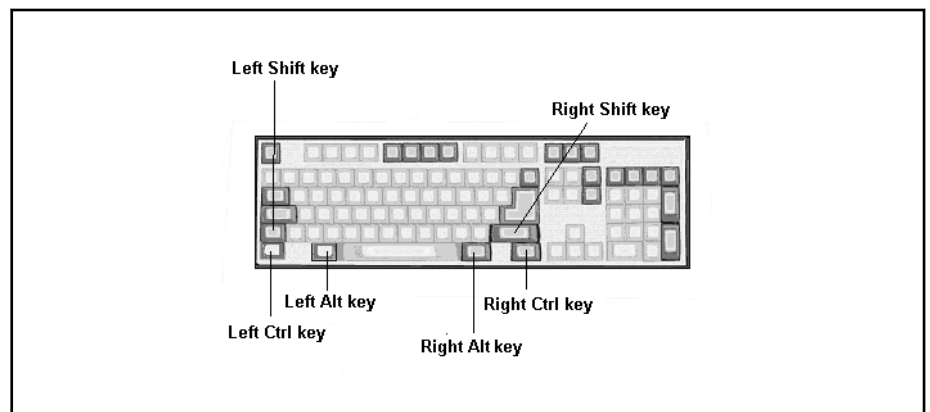


Fig. 5-29: Keyboard

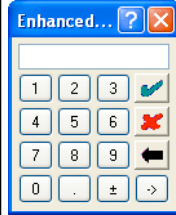
Virtual Keyboard

The virtual keyboard (VK) enables entering data (text or numbers) during runtime using a touchscreen device instead of a physical keyboard.

The virtual keyboard can be enabled for projects on the [Runtime Desktop dialog on page 69](#). A default keyboard type as well as a scaling (adjustment factor) can be specified for all virtual keyboards displayed at runtime.

When configuring objects and animations that support data input, a virtual keyboard can be assigned to the object by the virtual keyboard combo box (either the default or a specific one). The following default virtual keyboards are available:

Development Environment

Name	Description	
Alphanumeric	This virtual keyboard is used to enter alphanumeric values (characters and/or numbers).	
Enhanced...	This virtual keyboard is used to enter alphanumeric values (characters and/or numbers) on devices that are only equipped with a small display (e.g. PDAs). The "->" button enables scrolling through different panel labelings to find the required character	
Keypad	This virtual keyboard is used to enter numeric values (numbers only)	

There are two ways to start a virtual keyboard during runtime:

- Clicking on the object where a new value (new data) can be entered (e.g. the Text with Text Data Link animation configured with released input).
- Executing the build-in function [Keypad\(\)](#) on page 554.

The language of the virtual keyboard can be changed by the built-in [SetKeyboardLanguage\(\)](#) on page 629 function.

- New languages can be created for any virtual keyboard by editing the **VKlanguage.INI** file from the \BIN subfolder of "WinStudio", where **language** is the description of the language for the VK (e.g. **VKEN.INI** is virtual keyboards in English, **VKGE.INI** is virtual keyboards in German, etc.).

For further information on editing and configuring this file, contact the software vendor.

- By default, the VK is started close to the object it is connected with. However, the following settings in the project file can be configured (**project_name.app**) so that the virtual keyboard is always opened at a fixed position:

[Keypad] PosX=0 //Coordinate (in pixels) of the TOP coordinate where the virtual keyboard has to be started.

PosY=0 //Coordinate (in pixels) of the LEFT coordinate where the virtual keyboard has to be started.

Communication Tab

This tab is used to specify communication parameters relating to the project in general.

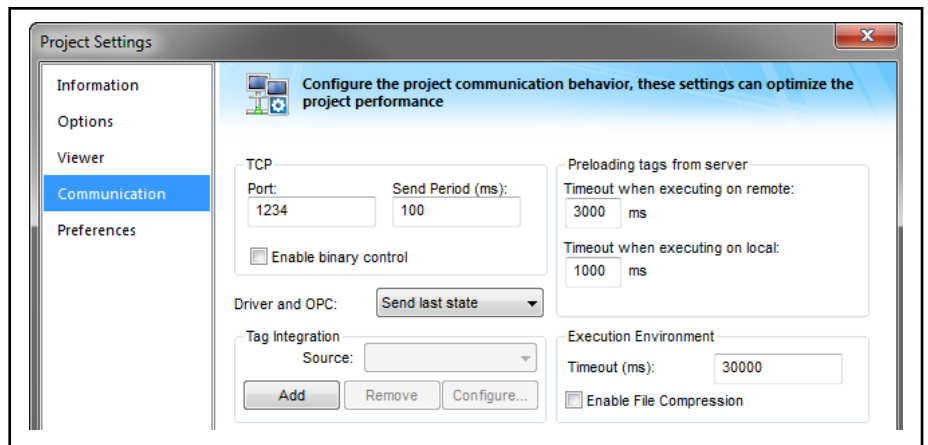


Fig.5-30: Project Settings: Communication tab

- **Driver and OPC:** Select the method used by all [communication drivers on page 333](#) and [OPC Client worksheets on page 342](#) configured in the current project when writing values to the remote PLC/device:
 - **Send every state:** When the communication task is configured to write values upon a change of a tag value, all changes in the tag value are buffered in a queue and sent to the device when the communication task (driver or OPC) is executed.
 - **Send last state:** When the communication task is configured to write values upon a change of tag value, only the current (last) value of the tag is sent to the device when the communication task (driver or OPC) is executed. When this method is selected and the tag value changed more than once while the communication task was executed, the transiently valid values of the tag are not sent to the device. This is the desired behavior for most projects.
- "TCP" area: In this pane, configure the communication settings for the [TCP/IP client and server modules on page 351](#):
 - **Port:** The TCP port used by the TCP/IP client and TCP/IP server modules on the current processor. If this value is changed in the local project, make sure to set the same value in a remote project communicating with the local one.



This setting is required for all projects and cannot be left blank. Make sure that the network configuration allows connections to this port on the station that is to be the project server.

- **Send Period (ms):** Period (in milliseconds) between two consecutive messages sent by the TCP/IP client or TCP/IP server modules to update tag values in the remote station. Typically, a lower value means better performance and higher traffic in the network.
- **Enable binary control:** Select this option to enable binary control when the TCP/IP client or the TCP/IP server module sends messages to the remote station. Binary control increases the safety of the system; however, it decreases the efficiency (performance) of the communication. When connecting two stations via the TCP/IP client and TCP/IP server module, make sure this setting is either enabled or disabled in both projects.
- "Execution Environment" pane: Use this item to configure the communication settings for the [Remote Management operation on page 61](#) to send the project files to a target system:

Development Environment

- **Timeout (ms):** Specifies the time (in milliseconds) that the project waits to communicate with the target system.
- **Enable File Compression:** Select this option to compress the system and project files before sending them to the target system. This may reduce the download time if there is a slow connection between server and target system. (If there is a fast connection, however, then selecting this option may "decrease" performance because each compressed file has to be decompressed on the target system before the next file is sent. Select this option only if there is an extremely slow connection, such as dial-up connection.) Data compression is disabled by default.
- "Preloading tags from server" pane: To improve performance, the runtime viewer loads all of the server tags of a screen to memory before it displays that screen. Configure the timeout settings for both remote and local viewers:
 - **Timeout when executing on remote** field: Specifies the time (in milliseconds) that a Secure Viewer or Thin Client running on a remote station waits to load the tags.
 - **Timeout when executing on local** field: Specifies the time (in milliseconds) that the viewer running on the local station (i.e. the server) waits to load the tags.
- **Tag Integration** pane: Use this item to connect and share tags on target devices. For further information, refer to ["Sharing tags on IndraLogic devices" on page 149](#).

Preferences Tab

Use this tab to configure personal settings.

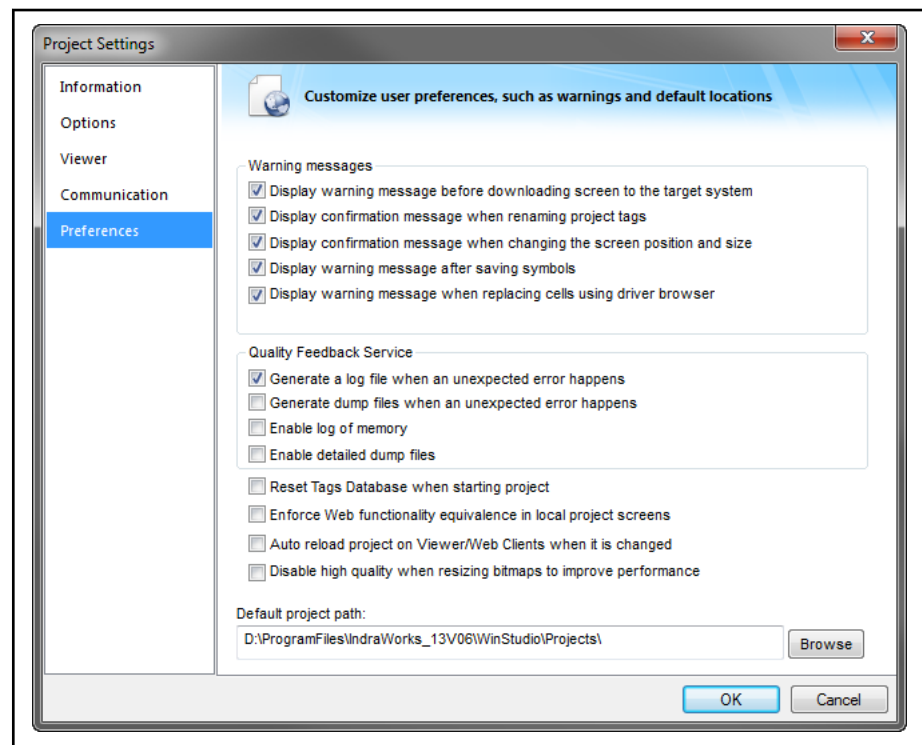


Fig. 5-31: Project Settings: Preferences tab

Warning messages

- **Display warning message before downloading screen to the target system:** If this option is selected and "WinStudio" is connected to a remote

Development Environment

station ([Remote Management on page 61](#) dialog), a prompt is displayed directly after saving a screen in the screen editor to download the updated screen to the remote station. If this option is disabled, the screen is downloaded automatically, regardless of any confirmation.

- **Display confirmation message when renaming project tags:** If this option is selected and the name of any tag is modified in the [Project Tags on page 106](#) worksheet, a prompt is displayed to replace the old tag name in the entire project by the new tag name. If this option is enabled, "WinStudio" executes the "global replace" command to replace the old tag name with the new tag name in all documents of the project (screens and worksheets).
- **Display confirmation message when changing the screen position and size:** If this option is enabled, a prompt is displayed to update the screen attributes (width, height, top and left) after modifying them on the <Layout> user interface.
- **Display warning message after saving symbols:** If this option is enabled, a warning message is shown after saving [symbols on page 200](#).

Quality Feedback Service

This pane enables configuration of projects to generate log files and/or dump files that can be used to diagnose hardware and software problems, such as memory leaks and unexpected errors. These files are saved in the \Web\Dump subfolder of the running project.

- **Generate a log file when an unexpected error happens:** If this option is enabled, the runtime modules append the log file (\Web\Dump\Dump.txt) whenever an internal exception (error) occurs. These exceptions may not necessarily crash the runtime modules, but they can affect the stability of the system and are to be investigated.



The log file is continually appended until it reaches its maximum size of 2MB. After it reaches its maximum size, the existing file is deleted and a new file is created.

- **Generate dump files when an unexpected error happens:** If this option is enabled, the runtime modules generate a new dump file (\Web\Dump*.dmp) with useful information about the conditions of the error. This is a binary file that can only be read by the software vendor.



Dump files are named WinXXX.dmp - where "XXX" is an identifying number (in hexadecimal format) automatically generated by the system - in order to prevent an existing file from being overwritten when a new error occurs. Therefore, if more than one error occurs, multiple dump files are saved in the directory. The log file indicates the name of the dump file associated with each error.

- **Enable log of memory:** If this option is enabled, runtime modules append the log file every 15 minutes with information about the current memory allocation. (The first log entry is written 15 minutes after the runtime module is started.) This information can be used to identify memory leaks.
- **Enable detailed dump files:** If this option is enabled, several details are logged that support analyzing the error.

Even if none of these quality feedback options are enabled, a post-mortem dump file (\Web\Dump\WinDump.dmp) is always generated to analyze the

Development Environment

crash when the runtime module is terminated by a fatal error. However, for debugging purposes, it is highly recommend to enable all options in this section and send the log file and all dump files to the software vendor.

Other preferences

- **Reset Tags Database when starting project:** If this option is enabled, project tags are always reset automatically when executing the runtime. For further information on this function, please refer to [Reset Tags Database on page 138](#).
- **Enforce Web functionality equivalence in local project screens:** If this option is enabled, the WinStudio software automatically puts out a warning if functions or features are selected that are incompatible with the remote runtime modules (e.g. Thin Client and Secure Viewer).



This option is disabled by default in order to maintain compatibility with previous versions of "Rexroth WinStudio".

- **Auto reload project on Viewer/Web Clients when it is changed:** If this option is enabled, remote stations (i.e. Thin Clients and Secure Viewers) check the server to see if they have the most recent version of the project. If this is not the case, the new version is downloaded automatically from the server.

Web Tab

The Web tab can be accessed via **WinStudio ► Tools ► Web Settings**.

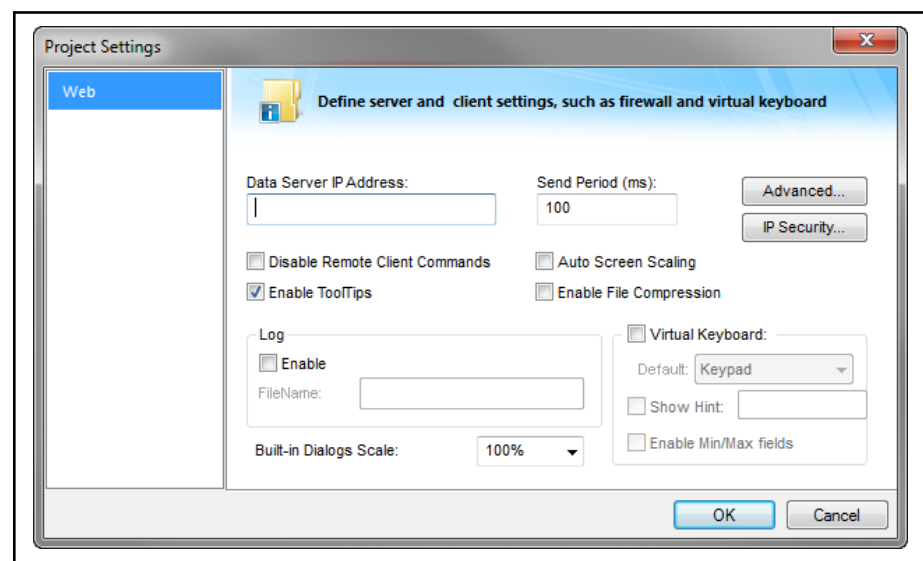


Fig.5-32: Project Settings: Web tab



If any of these settings are changed, [verify the project on page 59](#) before it is sent back to the target device.

The parameters on this tab are configured as follows:

- **Data Server IP Address:** Enter the IP address (or host name) of the data server station here. The data server station is the computer or device on which the TCP/IP server module of "WinStudio" is running. If this field is left blank, the Thin Client assumes that the web server (i.e. the address entered into the browser) is also the data server.



The IP address "127.0.0.1" (localhost) can be used to access the TCP/IP server on the local computer, regardless of its actual address in the network. This option is useful for local tests; however, the data server cannot be accessed from remote computers with this configuration.

- **Send Period (ms):** Enter a value to specify the interval (in milliseconds) used to exchange data between the server and the Thin Client stations. It means that the Thin Client sends a package with the new tag values to the server every "n" millisecond(s). The **Send Period** of the server is configured in the project settings (**Communication** tab in the Project Settings). The default value is "1000" (milliseconds). A lower value can be set in this field to increase the update rate between server and Thin Clients. This may result in higher data traffic in the network (the network is accessed more frequently) if the tags are changing continuously (faster than once per second).
- **Disable Remote Client Commands:** Enabled this checkbox to prevent that remote clients commands are sent from Thin Clients to the server. If this option is enabled, the Thin Client can read data from the server but cannot send data (tag values, command values) to the data server. In this case, the Thin Client station becomes a "read only" station.
- **Enable ToolTips:** Enable this checkbox to display the "ToolTips" configured for the screen objects when displaying them on a Thin Client (web browser).
- **Auto Screen Scaling:** Enable this checkbox, to automatically adjust the size of the screens displayed in a web browser window. Using this option, the screen adjusts to the size of the web browser window, regardless of its resolution.



The **Auto Screen Scaling** option is not valid for web browsers running under Windows CE operating systems.

- **Enable File Compression:** Enable this checkbox to compress the files stored in the "Web" subfolder of the project folder. This option is useful for reducing load time, particularly in case of a slow connection between server and Thin Client.
- **Log (Enable and Filename):** Enable this checkbox and enter a file name into the text field to generate a log file on the Thin Client station. This log file can be used for debugging.
- **Virtual keyboard:** If this option is checked, the [virtual keyboard on page 73](#) is enabled for projects running with Thin Clients. With this keyboard it is possible to enter data during runtime on touchscreens without entering data on a physical keyboard. For example, a [text object on page 175](#) the [Text Data Link animation on page 225](#) was applied to and the **input enabled** option was enabled.

A default configuration for the virtual keyboard can be set:

- **Default:** Select the default keyboard type to be used in the project if no keyboard type is specified by the calling object or function.
- **Scale:** Using this option, the keyboard size can be decreased or increased to adjust to the size of the target screen. A value of 100% represents the default size of each keyboard type.
- **Show Hint:** If this option is enabled, a hint is displayed in the title bar of the keyboard. For objects, the hint is configured in the [Ob-](#)

Development Environment

ject Properties dialog on page 164. Otherwise, enter a string or string tag in the **Show Hint** field as a default hint.

- **Enable Min/Max fields:** If this option is enabled, the permissible minimum and maximum values are displayed at the bottom of the keyboard. For objects, these values are configured in the [Object Properties dialog on page 164](#). For other cases, the [Min and Max properties on page 140](#) of the associated Tag are used by default.



The **Min** and **Max** fields are displayed only on the **Keypad** keyboard type, and only if the associated tag is defined as Integer or Real. If the Min value exceeds the Max value, the input is disabled. If Min/Max configured on the object is different from Min/Max configured in the tag properties, the project attempts to convert the input accordingly.

IP security settings

IP Security: Click this button to open the IP Security dialog.

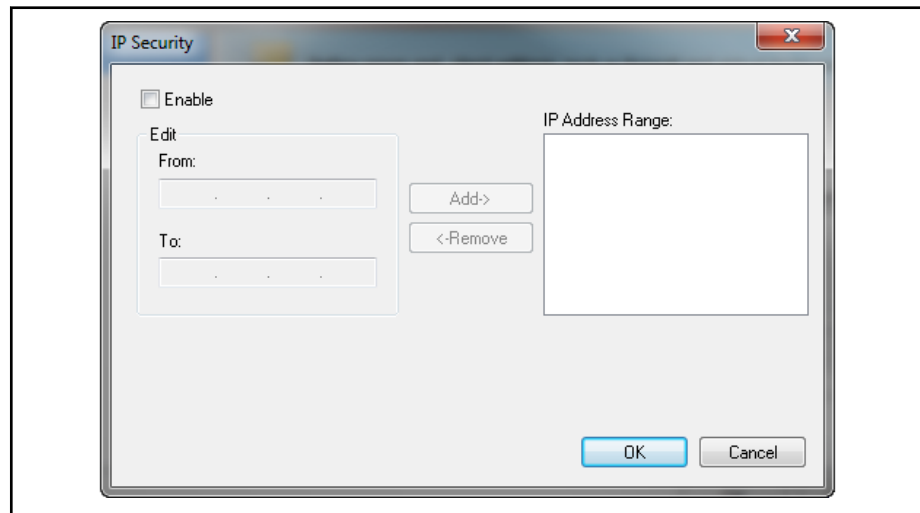


Fig. 5-33: IP Security dialog

The parameters of this dialog are used to specify the range of IP addresses for the computers that are allowed to access the project as Thin Clients. This option is useful if the IP Address of the Thin Client computers that are allowed to connect to the web server can be controlled.

Click the **Enable** checkbox and enter the IP addresses in the **From** and **To** fields in the active **Edit** pane to specify the IP address range. Use the **Add** and **Remove** buttons to move the IP addresses into the **IP Address Range** list. If a Thin Client attempts to connect to the server, it checks for an IP address for the Thin Client station that is within any range configured in the IP Security dialog. If none is found, the server refuses the attempt to connect to the Thin Client station.



By default, IP security applies only to Thin Clients connecting to the data server. IP security can also be implemented for [database synchronization between projects running on different stations on page 351](#). To do so, insert the following parameter into the project file (`project_name.app`):

[TCP]

UseWebIPSecurity=1

Advanced settings

Advanced: Click this button to open a dialog to edit the Advanced web settings. In most cases, these settings do not need to be modified. However, depending on the project architecture, it is possible to configure advanced settings.

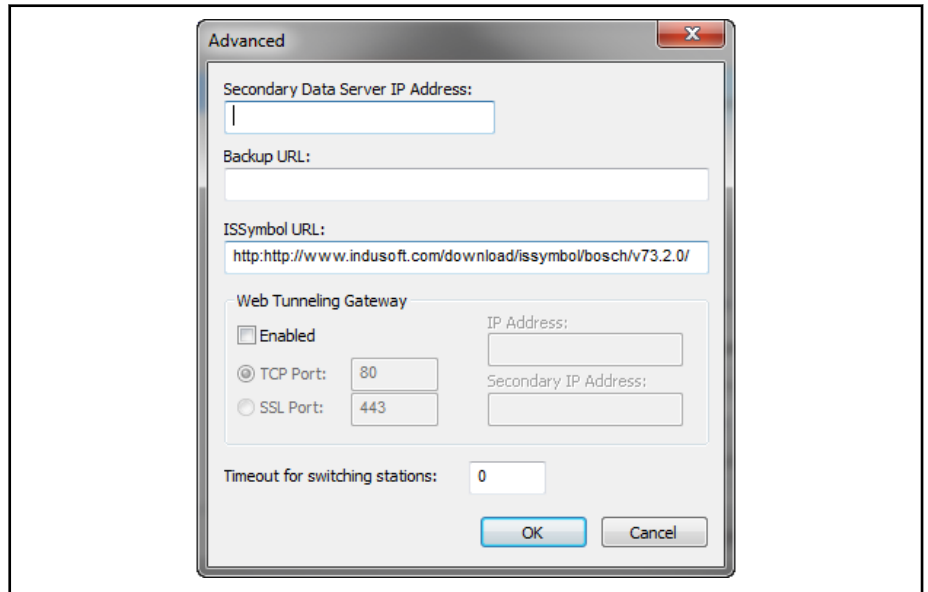


Fig. 5-34: Advanced dialog

- **Secondary Data Server IP Address** field: Enter the IP address (or host name) of the secondary data server station. The data server station is the computer or device on which the TCP/IP server module of "WinStudio" is running. This field has to be filled if data servers are to be used that are redundant from the Thin Clients. If the primary data server fails, the Thin Client attempts to connect to the secondary data server automatically.
- **Backup URL** field: Enter the URL where the web files are stored (files from the "Web" subfolder of the project folder). This URL is used to download the files from the secondary web server when the primary web server is not available.
- **ISSymbol URL** field: If the Thin Client connects to the server, it attempts to load the

[Control ISSymbol on page 397](#). If ISSymbol is not registered on the local computer (Thin Client), the browser attempts to download it from the URL specified in this field. The default URL is a web site where "Bosch Rexroth" keeps the latest version of ISSymbol available for download. A different location may need to be configured, especially if the Thin Client computer is not connected to the internet. The **ISSymbolVM.cab** (stored in the \BIN subfolder of "WinStudio") has to be available for the URL configured in this field.



If the Thin Client stations do not have access to the internet, it is recommended that the **ISSymbolVM.cab** file is made available to the web server station and that the respective URL is configured in this field.

Development Environment



Web browsers running under the Windows CE operating system cannot automatically download the Control ISSymbol (**ISSymbolCE.ocx**) from a remote location. **ISSymbolCE.ocx** has to be registered manually in the Windows CE device to be used as a Thin Client.

- **Web Tunneling Gateway:** Enable this option to activate the web tunneling gateway. Depending on the [architecture on page 399](#) of the project, the web tunneling gateway may be required to route the Thin Client computers to the data server.
- **TCP Port:** Select this option when using HTTP with the web server of Microsoft IIS. Specify the TCP port used by the HTTP web server here ("80" is the default TCP port for HTTP protocols).
- **SSL Port:** Select this option when using SSL (Secure Socket Layer) with the Microsoft IIS web server. The TCP port used by the HTTPS web server can be specified here ("443" is the default TCP port for HTTPS protocols).
- **IP Address:** IP address of the web server computer where the web tunneling gateway is running. This has to be the IP address of the web server accessible from the Thin Client station(s).
- **Secondary IP Address:** IP address of the Web server computer where the secondary web tunneling gateway is running. This has to be the IP address of the secondary web server accessible from the Thin Client station(s). This field has to be configured if redundant web servers are to be used.



Most of the web settings can be modified dynamically by the [Set-WebConfig on page 631](#) function. It is especially useful if the project is to be created just once and to make sure that the web settings are automatically configured when running the project on different stations with different IP addresses.

5.5 Tools

WinStudio offers various tools to support setting, configuration and analysis of WinStudio applications. These tools can be found in the context menu of the **WinStudio** folder in the **Tools** submenu.



Some functions are only available in a certain context. Please note the instructions for use in the descriptions of the individual tools.

- **WinStudio Library:** Opens the "WinStudio Symbols" tool window. Here, numerous preconfigured symbols can be found that can be used in WinStudio screens. Refer to [Symbols on page 199](#).
- **Reset Variables:** Resets all variables to their original values. Refer to [Tags Database Reset on page 138](#)
- **Variable Import Wizard:** Opens the "Variable import wizard" to apply WinStudio variables from other projects into the current application. Refer to [Importing from Other Project Databases on page 143](#)

Development Environment

- **Database Spy:** Opens the "Database Spy" tool window. This tool serves for displaying and modifying of variables in the tags database. Refer to [Database Spy on page 84](#)
- **LogWin:** Opens the output window in which the messages of the WinStudio runtime module of the local station can be observed. Refer to [LogWin Output Window on page 85](#)
- **Cross References:** Opens the "Cross References" tool window to search for the application of WinStudio variables in the current project. Refer to [Cross Reference List on page 87](#)
- **Add User:** Opens a configuration dialog to create users. Refer to [Creating and Configuring Users on page 374](#)
- This function is only available for the application type "WinStudio".
- **Insert Group:** Opens a configuration dialog to create user groups. Refer to [Creating and Configuring Groups on page 370](#)
- This function is only available for the application type "WinStudio".
- **Security Settings:** Opens the "Security Settings" dialog. In this dialog, the WinStudio user management can be activated and configured. Refer to [Using the Security System Configuration Wizard on page 359](#)
- This function is only available for the application type "WinStudio".
- **Log in / Log out user:** Opens the Log In dialog. Refer to [Logging On/Off on page 378](#)
- **Remote Database Spy:** Opens the input dialog to connect to other devices to display and modify their WinStudio variables. Refer to [Remote Database Spy on page 388](#)
- **Remote LogWin:** Opens the input dialog to connect to other devices to display their messages of the WinStudio runtime environment. Refer to [Remote LogWin on page 389](#)
- **Status:** Opens the "Execution Tasks" dialog. This dialog displays an overview of all tasks and their starting mode and current execution status.
- **Event Logger:** Opens the "Event Logger" dialog. Use this dialog to activate and configure logging of WinStudio messages. Refer to [Events and Event Logs on page 258](#)
- **Load Project:** Opens the Windows file browser. Already available WinStudio projects can be selected and applied to the current project

NOTICE

If a different WinStudio project is loaded, all files of the current WinStudio project is deleted.

In case of doubt whether to delete all existing files, cancel the process with the "Cancel" button.

- **Web Settings:** Opens the "Web Settings" configuration dialog. If the project is to be executed as web application, configure the web server in this dialog. Refer to [Web Tab on page 78](#)
- This function is only available for the application type "WinStudio".
- **Save all documents as HTML:** Creates or updates a HTML file for all WinStudio elements. Refer to [Creating a Simple Thin Client Program on page 393](#)

5.6 Screen/Worksheet Editor

Create or edit any kind of screens and worksheets for projects using the screen editor. Information can be input using the mouse and keyboard and control data can be output to processes and screens can be updated and animated based on the data input from processes.

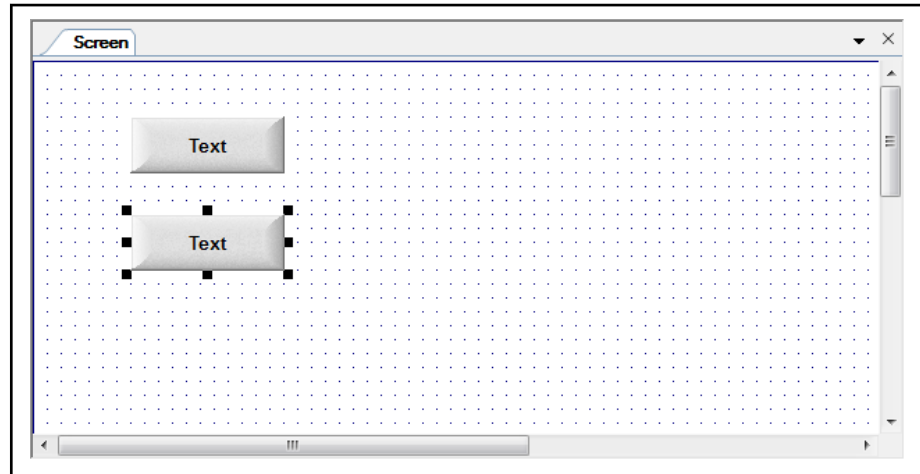


Fig.5-35: Screen/worksheet editor

Other screen editor features include:

- Point-and-click, drag-and-drop support for easy screen creation
- The option to group objects to keep their arrangement
- To edit group elements without ungrouping them
- Handling bitmap objects and background bitmaps

5.7 Database Spy

The "Database Spy" window is a tool for error analysis to display and change tag values and execute and test functions and mathematical expressions.

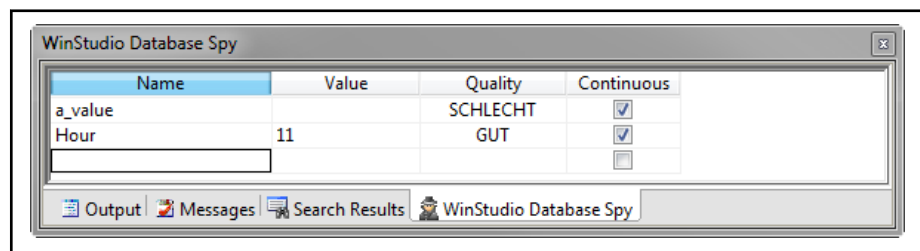


Fig.5-36: Example of Database Spy window

The window displays the following information:

- **Name:** Indicate an application variable, an internal variable or an expression to be monitored
- **Value:** Displays the value returned by the tag/expression
- **Quality:** Displays the quality (GOOD or BAD) of the value returned by the tag/expression
- **Continuous:** Select this option to have the project continuously evaluate the tag/expression

In order to monitor a variable, enter its name into the column or click on an empty field in the "Name" column and open the context menu. By means of

the "Insert variable or expression" function, the browser dialog can be opened alternatively and a variable can be selected.



The Database Spy is dockable, i.e. it can be moved to another location in the development environment.

Click on the title bar and drag the window to a new location.

Release the mouse button to attach or dock the window to its new location.

5.8 LogWin Output Window

Use the "WinStudio Log" window to display error analysis messages generated during project runtime. The window displays OPC, DDE, and TCP/IP transactions, module activations, trace messages, and so on.

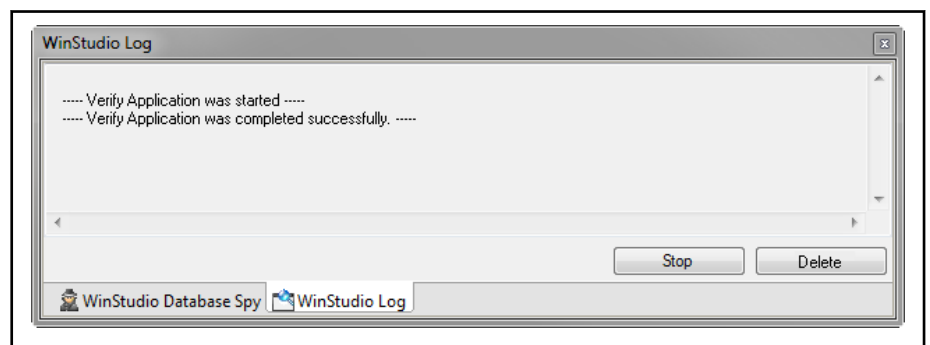


Fig.5-37: Example of output window

The "WinStudio Log" window is dockable, i.e. it can be moved to another location in the development environment. To do so, click on the title bar and drag the window to a new location. Release the mouse button to attach or "dock" the window to its new location.

Configuring the log to show additional information

By default, the log shows only error analysis and error messages — that is, messages indicating that the project is not running properly. If the log displays **all** messages created by "WinStudio", the output quickly becomes confusing.

To configure the messages displayed in the log, proceed as follows:

1. Right-click anywhere in the "WinStudio Log" window and then click **Settings** in the context menu. The "Log Settings" dialog is displayed:
2. On the "Log Options" tab of the dialog, select the specific types of messages that are to be displayed in the log.

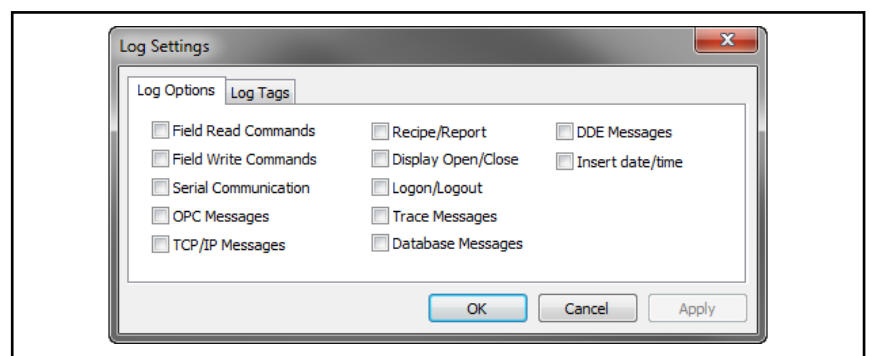


Fig.5-38: Log Settings - Options tab

Development Environment

Options	Description
Field Read Commands	Displays all field read commands that are sent to connected devices
Field Write Commands	Displays all field write commands that are sent to connected devices.
Serial Communication	Displays messages generated by configured device drivers on page 333 .
OPC Messages	Displays messages generated by OPC communications on page 342 .
TCP/IP Messages	Displays messages generated by TCP/IP communications on page 351 .
Recipe/Report	Displays messages of the tasks for recipe on page 319 and reports on page 320 .
Display Open/Close	Displays detailed information whenever a screen on page 156 is opened or closed: Disk Load Time: Time to load the screen file from the disk into memory Open Time: Time to load the WinStudio screen, including initialization tags used in the screen and running any "OnOpen" scripts or functions Total Load Time: Total time to load the screen (includes disk load time and open time above). "First Draw" Time: Time for the first drawing of screen objects "First OnWhile" Time: Time for running any "OnWhile" scripts or functions for the first time Total Open Time: Total time to open the screen (includes First Draw Time and First OnWhile Time as described above). Close Time: Time to close the screen, including finalizing tags used in the screen and running any "OnClose" scripts or functions Total Close Time: Total time to close the screen, including the time to write the screen file on the disk This information can be used to analyze runtime performance on low-end end systems. If a particular step of opening or closing takes an unusually long time, it can be identified and redesigned
Logon/Logout	Displays a message whenever a user logs on or logs out (For further information, refer to Security on page 357.)
Trace Messages	Shows messages generated by the Trace() function. This function is used to generate customized messages from the project to trace execution of the application
Database Messages	Displays messages generated by ODBC on page 321 and ADO.NET on page 328 database interfaces
DDE Messages	Displays messages generated by DDE communications on page 352 .
Insert date/time	Add a timestamp to each message

Tab.5-1: Log Settings - Options tab

3. In the "Log Tags" tab of the dialog, click **Add** to browse for [project tags on page 106](#).

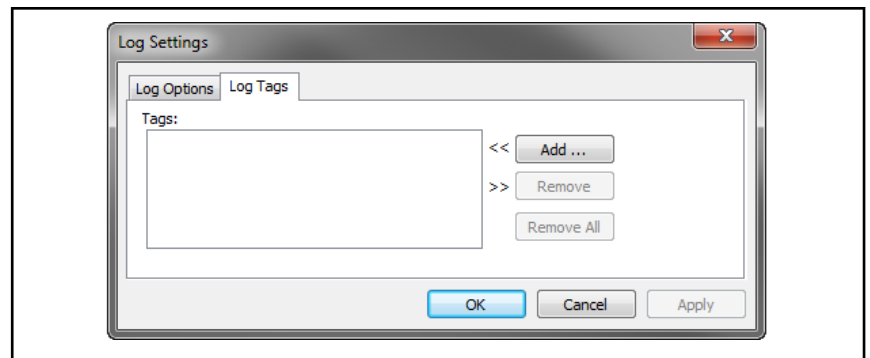


Fig.5-39: Log Settings — Tags tab

The Log pane of the LogWin window displays a message if the value of a selected tag changes.

4. Click **OK** to save the settings and close the "Log Settings" dialog.

5.9 Cross Reference List

If the information is required in which worksheets variables are used, open the the tool window: **WinStudio ► Tools ► Cross Reference**.

To search all worksheets, enter the name of the variable to be searched for in the **Tag Name** text field. As an alternative, open the variable browser via the



button to select a variable. The search is started after confirmation of the variable name by pressing the Enter key. Afterwards, all hits are listed in the log window (also refer to [output on page 85](#)).

By double-clicking a hit with the mouse, the respective worksheet is opened in the editor and the element using the searched variable is selected.

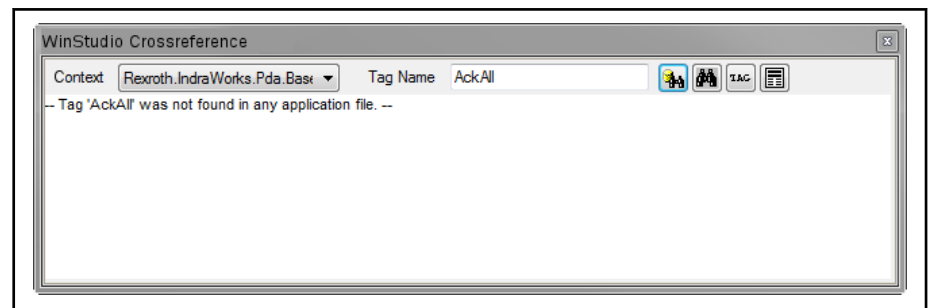


Fig.5-40: Cross reference search results



If a project consists of several visualization devices, select "Context" in the selection list to specify in which visualization device the WinStudio search is to be carried out.

5.10 Status Bar

The status bar can be found at the bottom of the development environment. It displays information on the active screen, e.g. the number of tags already used.

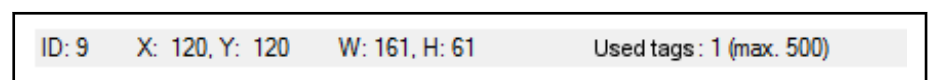


Fig.5-41: Example of status bar

The status bar fields (from left to right) are described in the following table:

Development Environment

Field	Description
Object ID	ID number of a selected screen object
Cursor position	The position of the cursor within the active screen or the active worksheet For screens, the position of the mouse cursor is given as X/Y coordinates, where X is the number of pixels from the left edge of the screen and Y is the number of pixels from the top edge of the screen to the cursor For worksheets, the position of the text cursor is given as line and column
Object size	The size (in pixels) of a selected screen object, where W is the width and H is the height
Used tags	Number of the tags already used in the project and the maximum possible number of variables for the current project type

Tab.5-2: Status bar fields

6 WinStudio Projects

6.1 Creating a New Project

6.1.1 Creating a New Project, General Information

WinStudio projects are always assigned to a visualization device in an IndraWorks project. To add a new visualization device to an open IndraWorks project, proceed as follows:

1. Add a visualization device to the IndraWorks project. To do so, the following options are available:
 - Open the **Library** window (in the "View" menu) and select the appropriate visualization type in the **Visualization** tab by clicking on it. Afterwards, keep the mouse button pressed and drag the selected visualization type to the project folder in the Project Explorer and release the mouse button.
 - Open the context menu of the project folder and select the visualization type of the visualization device in the **Add** submenu.

In both cases, the HMI Device Wizard opens.

2. In the HMI Wizard, select the appropriate entry from the "Device type" selection list.
3. In the "Name" input field, enter a text under which the device is to be displayed in the Project Explorer.
4. In the "IP Address" field, enter the computer name or the IP address of the device.
5. Select one of the following entries from the "Application type" selection list:
 - **WinStudio application:** Select this application type if an independent application is to be created that does not require additional visualization objects like F, M and OP panels or various standard applications (diagnostics, operating screens, etc.). The use of WinStudio-internal user management as well as support of Web clients is only available for this application type.
With this application type, the project is executed in the WinStudio runtime environment.
 - **HMI standard application:** Select this application type if panels and standard applications are to be used in addition to WinStudio objects. WinStudio is executed within the IndraWorks runtime.
 - **MTX standard application:** Select this application type if panels and MTX control-related standard applications are to be used in addition to WinStudio objects. WinStudio is executed within the IndraWorks runtime.
6. Under "Device version", select the version of the runtime environment that is installed on the visualization device.
7. Use the "Finish" button to confirm the settings. The HMI wizard creates a new folder for the visualization device in the Project Explorer. The WinStudio folder can be found under the Visualization folder:
 - With type **WinStudio application**, under the Visualization folder.
 - With type **HMI standard application** and **MTX standard application** under the "Screen application" folder.

WinStudio Projects

When creating a visualization device, the following settings are adjusted according to the device and application type:

- **Target platform:** For VEx devices with Windows CE operating system, "WinStudio CE Lite" is selected and "WinStudio Lite" for all other devices.
- **Resolution:** With WinStudio applications, the resolution corresponds to the size of the application window at runtime. This is usually the screen resolution of the target device (e.g. 1024x768). At HMI and MTX applications, this configuration shows deviations from the screen resolution as panels and headers are additionally displayed by the runtime environment. For this reason, the effective range for WinStudio screens is lower.
- **Viewer Task starting mode:** For WinStudio applications, the value is set to "Automatic". For HMI and MTX applications, the Viewer Task is started by the IndraWorks runtime environment, i.e. the value has to be set to "Manual".

6.1.2 Target Platforms and Product Types

Target platform and product type determine important factors about the project, such as how many tags the project supports and which features can be used at runtime.

Target platform	The target platform is generally either Windows or Windows Embedded.
Windows	A computer that runs one of the following operating systems: • Microsoft Windows XP Service Pack 3 or later • Microsoft Windows Vista Service Pack 1 or later • Microsoft Windows 7, all versions • Microsoft Windows Server 2003 Service Pack 2 or later • Microsoft Windows Server 2008, all versions • Microsoft Windows XP Embedded Service Pack 3 • Microsoft Windows Embedded Standard 7 (2009) The computer requires the installed full "Rexroth WinStudio" software, even if it is not used for project development, because the full software includes the modules that are needed to host the project runtime on Windows. The computer also requires an appropriate runtime license key. For further information, refer to Installation on page 37. If the target platform is Windows, all project features are available.
Windows Embedded	An embedded device that runs one of the following operating systems: • Microsoft Windows XP Embedded Service Pack 3 • Microsoft Windows Embedded Standard 7 (2009) • Microsoft Windows Embedded Compact (formerly known as Windows CE or Windows Mobile), version 5.0 or later The device requires either EmbeddedView (for Windows XP Embedded and Windows Embedded Standard) or CEView (for Windows Embedded Compact) to be installed. The device does also require an appropriate runtime license key. For further information, refer to Installation on page 37.

Please note: If the target platform is **Windows Embedded**, certain project features are not available:

- All DDE features, including the DDE Server task and the DDE Client task and worksheets.
- ODBC Runtime task, worksheets, and functions. (ADO database interface is still available.)
- OPC HDA Server task.
- Minor graphic features such as support for image formats other than BMP or JPG and fill effects for shapes other than rectangle.
- Miscellaneous other built-in functions, as described in the documentation for the internal Scripting Language.



In most cases, the following applies: If the project runtime is to be hosted on a Windows XP Embedded or Windows Embedded Standard device and project development is not to be performed on that device, only EmbeddedView is to be installed on the device and **WinStudio Embedded** selected as target platform. EmbeddedView has a smaller footprint than the full "Rexroth WinStudio" software, and can be installed and managed remotely.

However, if any of the features that are not available when the **WinStudio Embedded** target platform is to be used, the full "Rexroth WinStudio" software has to be installed on the device and **WinStudio** has to be selected as target platform.

The full "Rexroth WinStudio" software cannot be installed on a Windows Embedded Compact device under any circumstances.

Product type

The product type primarily determines how many tags can be created in the project, including tags shared or imported from other systems. The "WinStudio" project type supports considerably more tags than "WinStudio Embedded".

6.1.3 Changing the Target System of an Existing Project

On adding a visualization device, the target system is preconfigured according to the respective device type. The default settings have to be changed under the following circumstances:

- More than 500 tags are required for the application
- At least one of the following functions is to be used: Trends, Recipes, Reports, External Databases
- The WinStudio-integrated user management or web functionality are to be used with the "WinStudio application" application type

If this does not apply to the project, the target system has to be changed to WinStudio 1.5k (for 1500 tags) or higher. All target systems above and including "WinStudio 1.5k" enable using the above mentioned functions.

To select "WinStudio 1.5k" or higher as target system, a respective WinStudio Engineering license is required. Otherwise, this target system cannot be selected.

To change the target system with a suitable WinStudio Engineering license, proceed as follows:

1. Open the WinStudio properties dialog.
2. Click on the "Options" tab.

WinStudio Projects

3. Open the selection list in the **Target system** pane. A list of available product types is displayed. Select the new product type of the project.

For further information, please refer to [Target Platforms and Product Types on page 90](#).

4. Close the dialog with "OK".

The project is converted to the selected product type.



By changing from a WinStudio target platform to a WinStudio Embedded target platform, some features become unavailable.

6.2 E-Mail Configuration

Some features, such as alarms and certain functions, can be used to send E-Mails to designated recipients. To use these features, the project E-Mail settings have to be configured.

The E-Mail settings can be configured at any point at runtime by means of the "CnfEmail" function. However, default settings can be configured that are automatically used if the project is run for the first time and then restored as required at runtime, overwriting any changes made by calling the "CnfEmail" function.

1. Open the context menu of the "WinStudio" folder and select the entry **Tools ► E-Mail and FTP configuration**. The "E-Mail and FTP configuration" dialog is displayed.
2. Click the **E-Mail Settings** tab.

Fig.6-1: E-Mail Settings tab

3. In the **E-mail (From)** field, enter the E-mail address of the sender.
4. In the **Server** and **Port** fields, enter the server address and port number of the outgoing mail server. The default port for SMTP is 25, however, it may vary according to the server and network configuration. Please contact your email provider.
5. If the outgoing mail server requires authentication, select **My server requires authentication**. If the authentication has to be encrypted additionally, select **Enable SSL**. Enter the access data into the **User Name** and **Password** fields.

Most outgoing mail servers require authentication to prevent spamming and other abuse from unauthorized users.



Encryption via SSL is not supported in projects running on Windows CE target systems.

6. In the **Status** field, type the name of a tag (Integer type) that is to receive status codes when the project sends an E-Mail.
7. In the **Reload** field, enter a tag/expression. If the value of this tag/expression changes, the project reloads these default E-Mail settings.
8. Click **OK** to apply the changes and close the dialog.

6.3 FTP Configuration

WinStudio functions enable transmission of files between computers and FTPs. To enable these features, the project FTP settings have to be configured.

The FTP settings can be configured at any point at runtime by calling the [CnfFTP on page 687](#) function. However, default settings can also be configured that are automatically used if the project is run for the first time and then restored as required at runtime, overwriting any changes made by calling the **CnfFTP** function.

1. Open the context menu of the "WinStudio" folder and select the entry **Tools ► E-Mail and FTP configuration**. The "E-Mail and FTP configuration" dialog is displayed.
2. Click the **FTP Settings** tab.

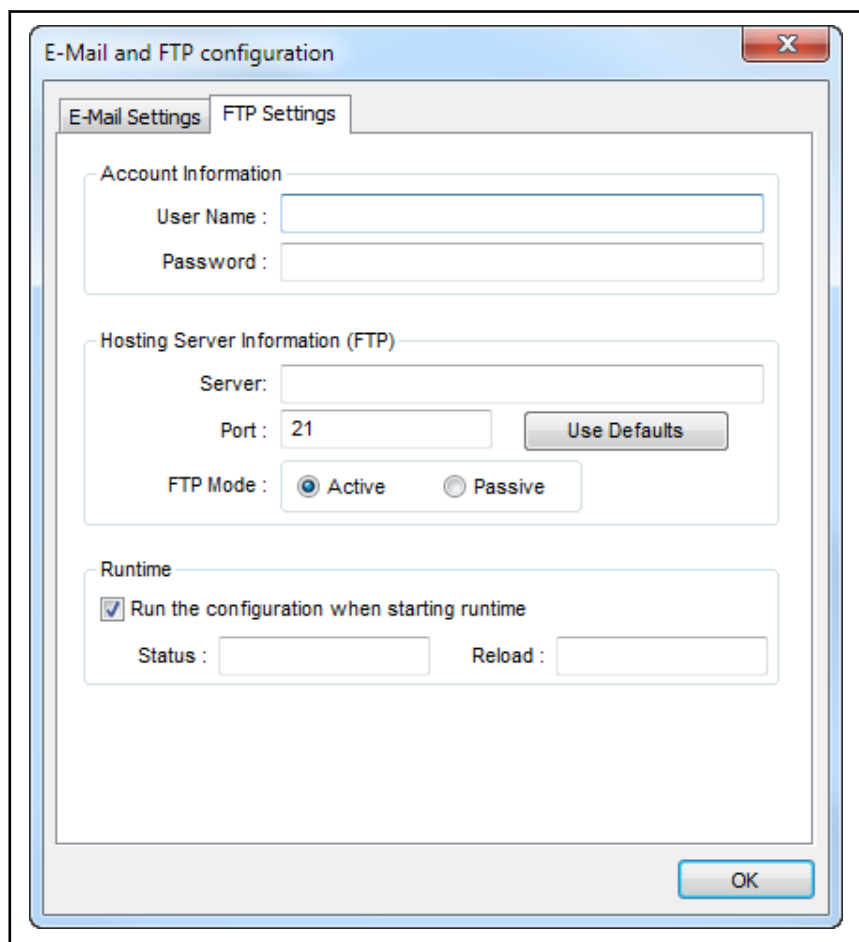


Fig.6-2: FTP Settings tab

3. Enter the access data for the FTP server into the **User Name** and **Password** fields.
4. In the **Server** and **Port** fields, enter the server address and port number. The default port for FTP is 21, however, it may vary according to the server and network configuration. Please contact the server administrator.
5. Select **Active** or **Passive** mode, depending on the configuration of the server. Use the passive FTP mode to bypass some network firewalls. Again, please contact the server administrator.

6. In the **Status** field, enter the name of a tag (Integer type) that is to receive status codes if the project transfers a file.
7. In the **Reload** field, enter a tag/expression. If the value of this tag/expression changes, the project reloads these default FTP settings.
8. Click **OK** to apply the changes and close the dialog.

6.4 Project Execution under Windows Services

"WinStudio" projects can be configured to run under "Windows services". Microsoft Windows services, formerly known as NT services, enable the creation of long-running programs that run in their own Windows sessions. These sessions can be automatically started when the computer boots, can be paused and restarted, and are not displayed on any user interface. These features make services ideal for use on a server or whenever long-running functionalities are required that do not interfere with other users who are working on the same computer. Services can also be executed in the security context of a specific user account that is different from the logged-on user or the default computer account. For further information about services, please refer to the [Microsoft Developer Network \(MSDN\) Library](#).

Why are projects to be executed under Windows services?

- To ensure that the project always runs with whatever system privileges it needs, regardless of the privileges of the user that is currently logged on to Windows
- To prevent the user from interfering with the project while it is running
- To let the project keep running if there is no user logged on at all

Creating and configuring the Windows service



All of the procedures described below were tested using Microsoft Windows XP Service Pack 3. To perform these actions, the logged in user requires administrative privileges and should know how to use the "Computer Management" console. (To access the console, right-click the **My Computer** icon and choose **Manage** from the shortcut menu.)

There are two ways to create and configure the Windows service for the project: The "Service Configuration" tool in WinStudio itself or the command-line utility program.

Service Configuration Tool

New services can be configured and executed in the development environment by selecting **Service Configuration** in the WinStudio context menu. The "Service Configuration" dialog is opened:

WinStudio Projects

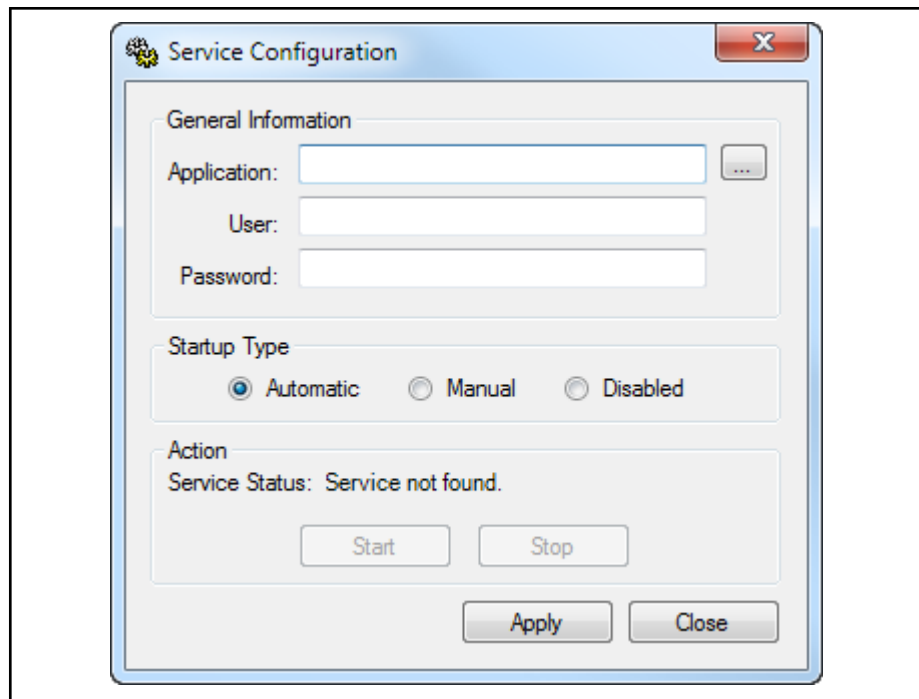


Fig.6-3: Service Configuration dialog

Application field

The location of the project file (**project_name.app**) that the service loads and runs when it is started. A complete file path has to be entered.

User field

The Windows user account under which the service is to be executed. This is an optional setting. If it is not used, the service runs under Local System.



Try to avoid running the service under Local System. This account has too many rights for the file system and too few rights to properly execute [OPC Client and Server modules on page 342](#). The best alternative is to create a user solely to run "WinStudio" and configure its rights to fit the project requirements. For more detailed information, refer to "Configuring User Rights" below.

Password field

The password for the specified user account. This is an optional setting. It is not required if no user is specified or if the specified user does not have a password.

Startup Type field

How the Windows service starts. The following options are available:

- **Automatic:** The service starts automatically when the computer boots.
- **Manual:** The service can be started manually in the "Computer Management" console or by clicking **Start**, as described below.
- **Disabled:** The service is created and then disabled. It cannot be executed before an administrative user enables it in the "Computer Management" console.

Action pane

Starting or stopping the service. Please note: These buttons are not enabled before the service is actually created.

Creating a New Service:

1. Next to the **Project** field, click ... to open a standard Windows file browser. Use the browser to select the project file.
2. In the **User** and **Password** fields, enter the username and password (if any) for the Windows user account under which the service is to run.
3. Select a **Startup Type**.
4. Click **Apply**. The service is created with the specified settings.

After the service has been created, it appears in the "Services" console (**Start ► Control Panel ► Administrative Tools ► Services**) under the name "Rexroth WinStudio". Use this console to quickly stop and restart the service if the "WinStudio" development environment is not to be executed

Command-line Utility

The service can also be configured by using the command-line utility **StdSvcInst.exe**. It offers a few more options than the Service Configuration tool described above – such as specifying a name and description for the service – and it can be used without launching the "WinStudio" development environment. The utility is located in the **\Bin** folder of the "WinStudio" program directory. To run the utility, open a command prompt, navigate to the **\Bin** folder, and enter the command with the desired options.

The utility has the following command syntax:

StdSvcInst { -create -app filepath -startup { auto | manual | disabled } -user username -password password -name displayname -descr description | -start | -stop | -delete }

-create

Creates the Windows service.

-app filepath

Specifies the project file (**project_name.app**) that is loaded and executed if the service is started. (This is the same as the **Project** field in the "Service Configuration" dialog.) The complete file path has to be specified and enclosed in quotation marks.

This command line parameter is "required" to create a new service.

WinStudio Projects

-startup { auto | manual | disabled }

Specifies how the service starts. (Corresponds to the **Startup Type** field in the "Service Configuration" dialog.) This command line parameter is optional. If it is not used, the default behavior for a new service is "manual".

-user username

Specifies the Windows user account under which the service runs. (Corresponds to the **User** field in the "Service Configuration" dialog.) This is an optional command line parameter. If it is not used, the service runs under Local System.

-password password

Specifies the password for the given user account. (Corresponds to the **Password** field in the "Service Configuration" dialog described above.) This is an optional parameter. It is not needed if no user is specified or if the specified user does not have a password.

-name displayname

Specifies the service name that is displayed in the "Computer Management" console. The name has to be enclosed in quotation marks. This is an optional command line parameter, the default name is "WinStudio".

-descr description

Specifies the service description that is displayed in the "Computer Management" console. The description has to be enclosed in quotation marks. This is an optional command line parameter.

-start

Starts the service. This corresponds to starting the service using the "Computer Management" console or by clicking **Start** in the "Service Configuration" dialog.

-stop

Stops the service. This corresponds to stopping the service using the "Computer Management" console or by clicking **Stop** in the "Service Configuration" dialog.

-delete

Deletes the service.

Example: Creating the Service

In this example, a new Windows service is to be created with the following options:

"WinStudio" Project File	C:\Users\username\My Documents\WinStudio v7.3 Projects\project_name\project_name.app
Startup Mode	Automatic
User	"WinStudio"

WinStudio Projects

Password	"WinStudio"
Service Name	"Rexroth WinStudio"
Service Description	"Starts a WinStudio project"



Please note: The system requires a "WinStudio" user account with password "WinStudio".

To create the service with the required options, proceed as follows:

1. Make sure to be logged in as a user with administrative rights.
2. Open a command prompt (**Start ► All Programs ► Accessories ► Command Prompt**)
3. Navigate to the Bin folder: **cd Bin**
4. Enter the following command:

```
StdSvclnst -create -app "C:\Users\username\My Documents\WinStudio
v7.3 Projects\ project_name/project_name.app" -startup auto -user
WinStudio -password WinStudio -name "WinStudio" -descr "Starts a
WinStudio project"
```

If the procedure is successful, the system displays the message "Service created". Otherwise, an error message is displayed.

Example: Changing the Project File

After creating the service, the "WinStudio" project file that it runs may need to be changed. To do so, use the **-app** command line parameter:

1. Make sure to be logged in as a user with administrative rights.
2. Stop the service if it is running.
3. Open a command prompt.
4. Navigate to the Bin directory.
5. Enter the command, for example, to set NTDemo as the project file:

```
StdSvclnst -app "C:\Program Files (x86)\Rexroth\WinStudio\Demos\
NTDemo\NTDemo.app"
```

Example: Deleting the Service

To delete the service, proceed as follows:

1. Make sure to be logged in as a user with administrative rights.
2. Stop the service if it is running.
3. Open a command prompt.
4. Navigate to the Bin directory.
5. Enter the following command:

```
StdSvclnst -delete
```

Configuring User Rights

The service runs under the rights of the user account specified in the **User** field of the "Service Configuration" tool (or by the **-user** switch of the command-line utility). If "WinStudio" requires a system resource for which no user rights are available, it fails. For this reason, the account needs to be configured to have the necessary rights.



The following actions can only be performed by a user with full administrator rights.

WinStudio Projects

Enabling "Log on as a service" for a user account

Before anything else, the specified user account has to be enabled to log on to the computer as a service. To enable the account:

1. Open the "Local Security Settings" console (**Start ► Control Panel ► Administrative Tools ► Local Security Policy**).
2. In the console window, select the folder "Security Settings\Local Policies\User Rights Assignment".
3. In the list of available policies, double-click **Log on as a service**.

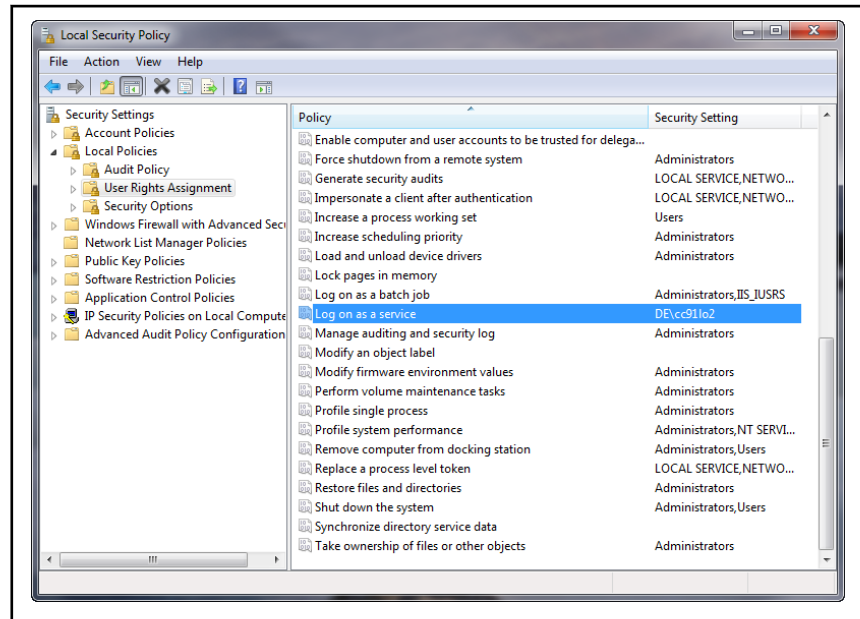


Fig.6-4: Selecting the "Log on as a service" policy

The "Log on as a service" properties page is displayed.

4. Click **Add User or Group**. The "Select Users or Groups" dialog is displayed.
5. Enter the name of the user account under which the service is to run.
6. Click on **OK**.

Giving the user account full control over the program directory

For the "WinStudio" project to run properly, the specified user account requires full control over the program directory and all of the files in it. To give the account those privileges, proceed as follows:

1. Open "Windows Explorer" (**Start ► All Programs ► Accessories ► Windows Explorer**).
2. Select **Tools ► Folder Options**. The "Folder Options" dialog is displayed.
3. Click the **View** tab.
4. Make sure the option **Use Sharing Wizard** is selected.

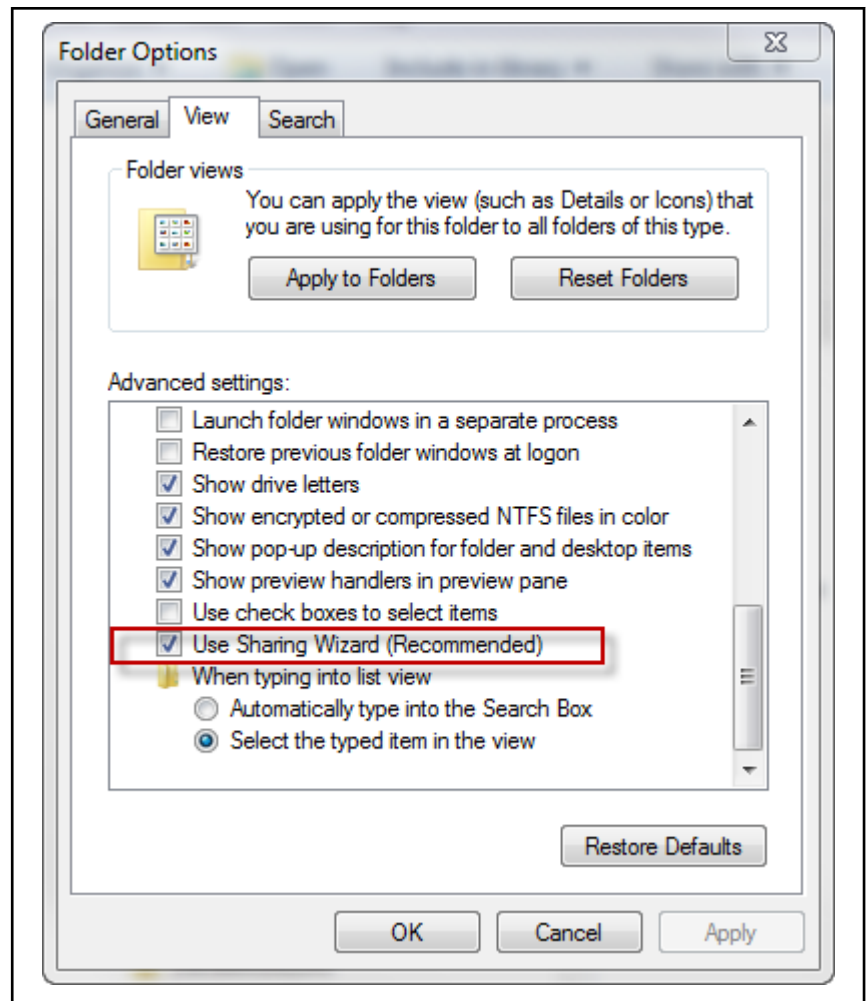


Fig. 6-5: Make sure the option "Use Sharing Wizard" is enabled.

5. Click **OK** to close the "Folder Options" dialog.
6. Still in "Windows Explorer", locate and select the "WinStudio" program directory, i.e., the folder that contains the file **project_name.APP**.
In this example, the folder is "NTDemo".
7. Right-click the folder and choose **Properties** from the shortcut menu. The Properties dialog of the directory is displayed.
8. Click on the **Security** tab.

WinStudio Projects

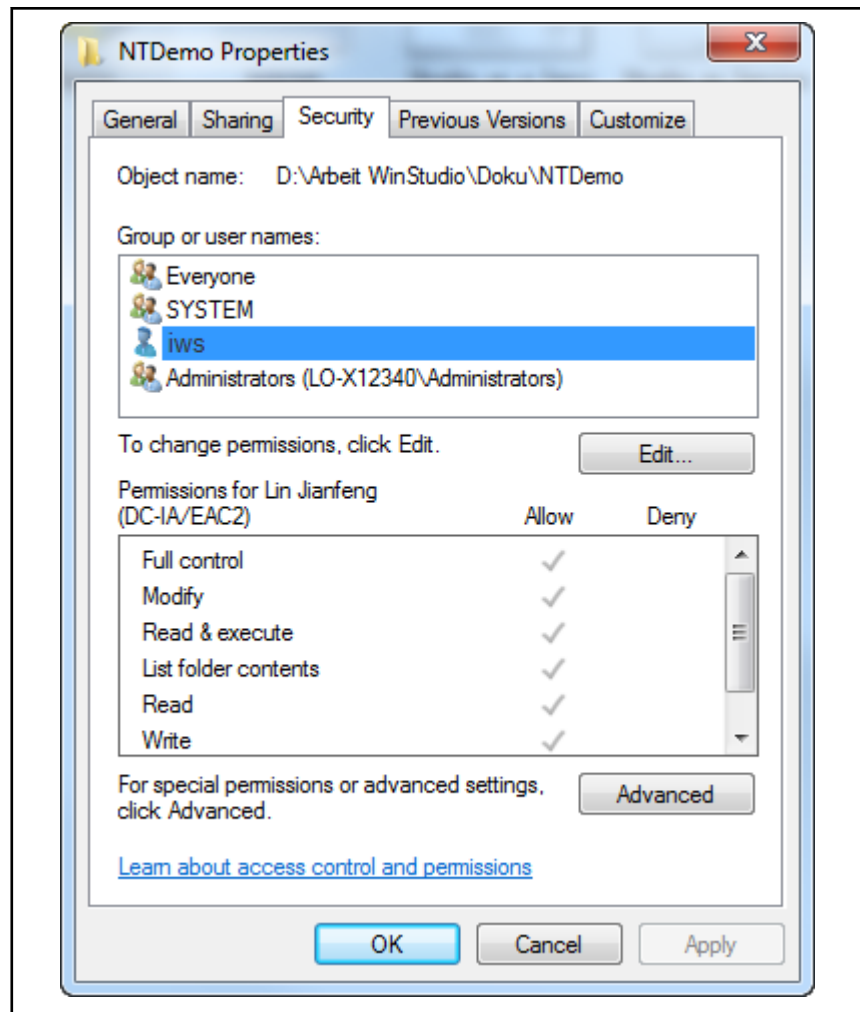


Fig.6-6: Properties dialog for the program directory

9. Click **Add** and then add the user account that was specified when creating the service. In this example, the user is "iws".
10. Select this user and then set its "Full Control" permission to **Allow**.
11. Click **OK** to apply the changes and close the dialog.
12. Close the "Windows Explorer".

Allowing the user account to run the OPC Client/Server module

As mentioned previously, normal users have too few rights to properly execute the OPC Client/Server module. Therefore, the user account has to be configured to have these rights:

1. Open the "Component Services" console (**Start ► Control Panel ► Administrative Tools ► Component Services**).
2. In the console window, select the folder **Console Root\Component Services\Computers\My Computer\DCOM Config**.

WinStudio Projects

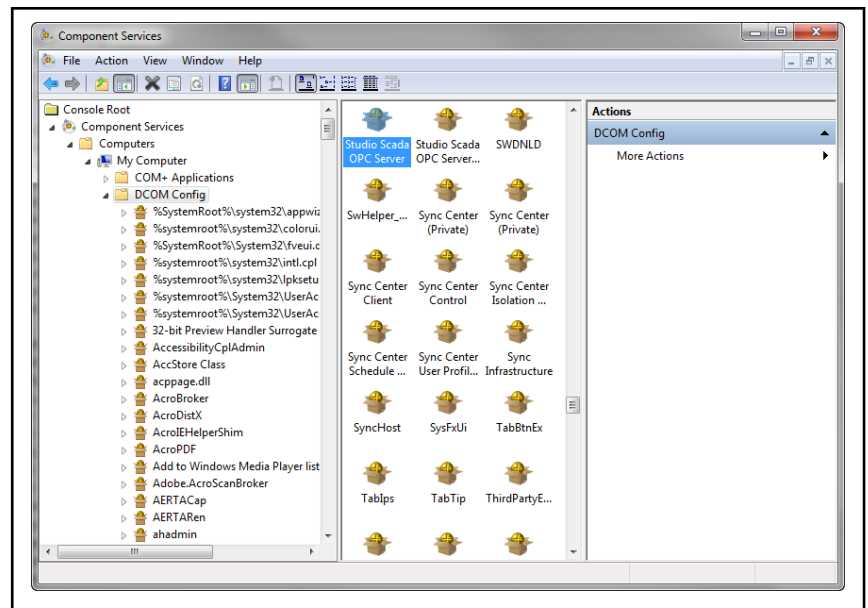


Fig.6-7: Selecting the OPC Server module in the Component Services console

3. In the DCOM Config pane, right-click the **WinStudio Scada OPC Server** icon and then choose **Properties** from the context menu. The "WinStudio SCADA OPC Server Properties" sheet is displayed.
4. Click the **Identity** tab.
5. Click **This user** and then complete the fields with the same user and password that were specified when the service was created.
In this example, the user is "iws" and the password is also "iws".

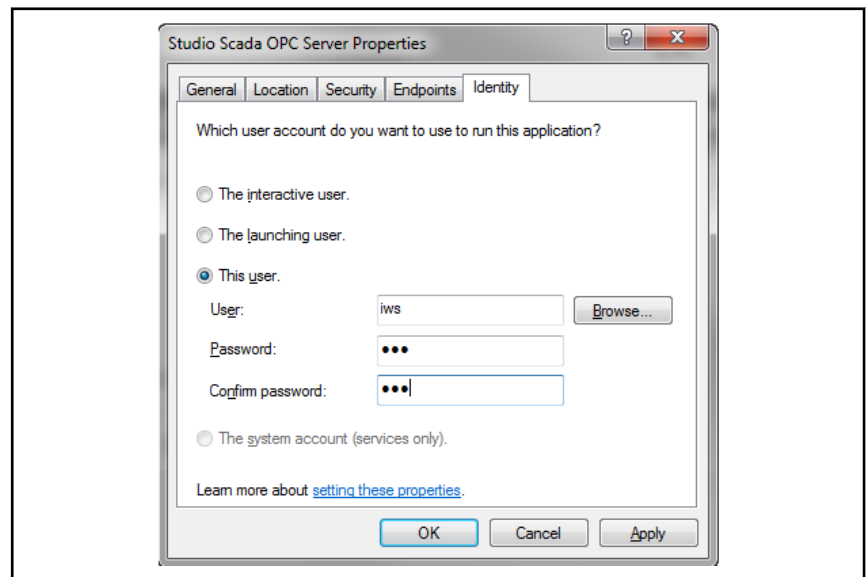


Fig.6-8: Entering user name and password

6. Click **OK** to apply the changes and close the property sheet.
7. Close the "Component Services" console.

Troubleshooting

If "WinStudio" is running under Windows services, it has no user interface. For this reason occurring errors will be logged as Windows application events only. Check the messages by using the "Event Viewer" console (**Start** ► **Control Panel** ► **System and Security** ► **Administrative Tools** ► **Event Viewer**).

WinStudio Projects

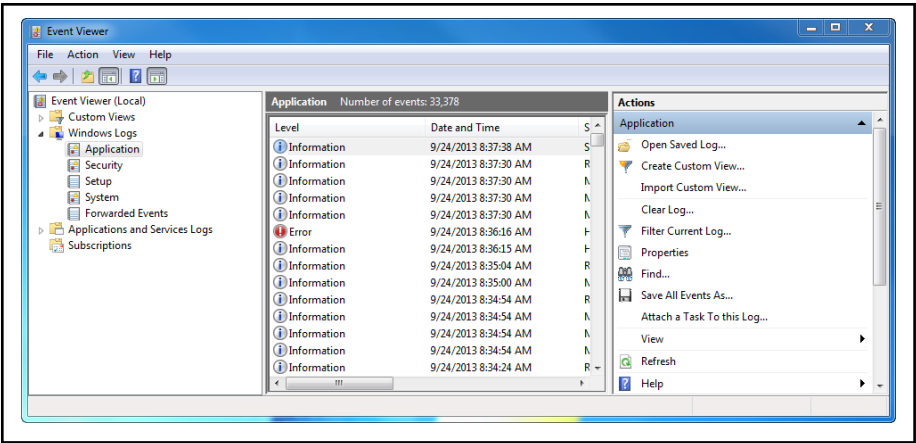


Fig.6-9: WinStudio error messages in the Event Viewer

7 Working with Tags

7.1 Tags and the Project Database

7.1.1 Tags and the Project Database, General Information

Tags are a core component of any "WinStudio" project. In simple terms, tags are variables used by "WinStudio" to receive and store data obtained from communication with plant floor devices, from the results of calculations and functions, and from user input. Additionally, tags can be used to display information on screens (and Web pages), to manipulate screen objects, and to control [Runtime Tasks on page 315](#).

However, tags are more than simple variables. "WinStudio" includes a real-time database manager that provides a number of sophisticated functions such as time-stamping of any value change, checking tag values against run-time minimum and maximum values, comparing tag values to alarming limits, and so on. "WinStudio" tags have both a value as well as various properties that can be accessed - some at development and others only at runtime.

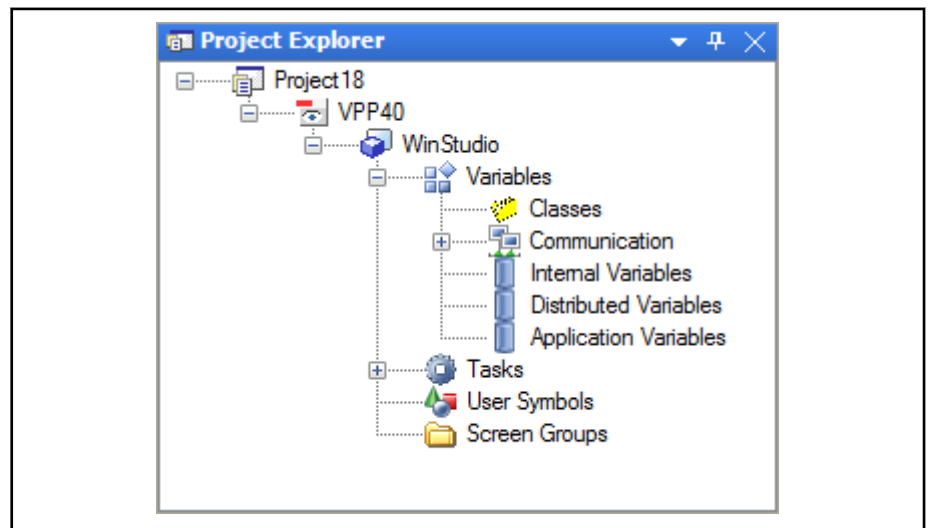


Fig.7-1: WinStudio tags in the Project Explorer

All WinStudio tags are assigned to one of the following categories that are respectively illustrated as elements in the "Variables" folder:

- [Application Variables on page 106](#): are tags created during project development. For example, application variables are used at the following positions:
 - Screen tags
 - Tags reading values of devices or writing into a device
 - Control tags
 - Auxiliary tags used to perform mathematical calculations
- [Distributed Variables on page 109](#): Tags are created in a PC-based control program and then imported into the "WinStudio" Tags database. Shared tags cannot be modified within "WinStudio" – the tags have to be modified in the original PC-based control program and then re-imported into the Tags database.
- [Internal Variables on page 110](#): are predefined tags with defined functions that are used for "WinStudio" supervisory tasks. For example:
 - Date tags hold the current date in string format

Working with Tags

- Time tags hold the current time in string format

Most internal variables are "read-only". This means, these tags cannot be added to the database, edited, or removed from the database.

To display a list of internal variables, open the "Internal Variables" editor in the "Variables" folder. The preceding figure illustrates a partial list of internal variables.

After creating a tag, it can be used anywhere within the project. The same tag can be used for more than one object or attribute.

7.1.2 Application Variables

Application Variables, General Information

The "Application Variables" editor displays all created tags of the current project. Application Variables can be applied to display values, to read and write at external devices, to control and to perform mathematical calculations, etc.



Prior to deleting a tag, it is highly recommended to use the **Cross reference search**  tool to ensure that the tag is not used at a different position in the project (screens, math worksheets, etc.). If a tag is deleted from the project database that is being used in another part of the project, this causes a compiling error and the project to function improperly.

To create a new tag, right-click on the "Application Variables" element and select **Add variable** from the context menu.

The "New Tag" dialog is displayed, as illustrated in the following figure:

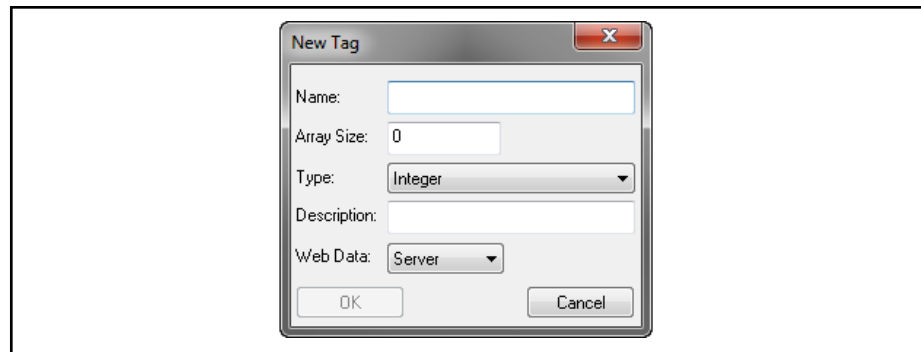


Fig. 7-2: New Tag dialog

Use this dialog to specify the following parameters.

- **Name:** Enter a name for the new tag. The first character has to be a letter and the name can contain up to 255 characters.
- **Array Size:** Enter a value to specify the size of the tag. Any size greater than 0 implies that the tag is an [Array on page 114](#).
- **Type:** Select a standard [Tag Type on page 113](#) from the list (**Boolean**, **Integer**, **Real** or **String**). Also new types, i.e., [Classes on page 108](#) can be defined from standard types.
- **Description:** Enter a tag description for documentation purposes.
- **Scope:** Click to select one of the following options:
 - **Local:** The information in this tag is not to be shared over the Internet.
 - **Server:** The information in this tag is to be shared over the Internet.

Working with Tags

These options have no effect on projects that do not have Web capabilities. If the **Scope** option for a project with Web capabilities is selected, any object property using "Local" will not work properly over the Web.

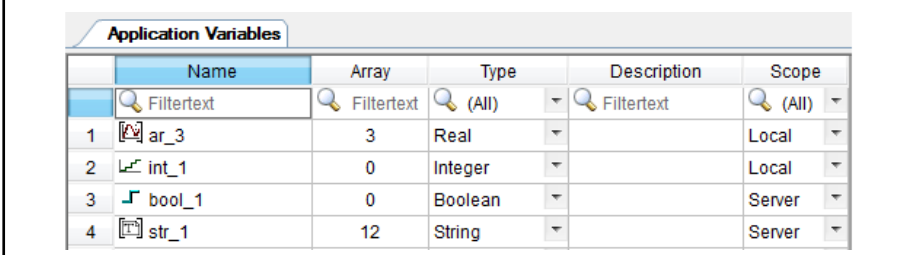


Tag names have to be unique. Tags that use the name of an existing tag cannot be created.

Tag properties can be displayed or edited in the following dialogs:

- "Tag Properties" dialog: Open the "Cross References" tool window, enter the tag name and press the **Tag Properties** button.
- "Application Variables" editor: In the **Variables** folder, click on the **Application Variables**  element.

The "Application Variables" datasheet contains five columns (refer to the following sample dialog).



	Name	Array	Type	Description	Scope
	 Filtertext	 Filtertext	 (All)	 Filtertext	 (All)
1	 ar_3	3	Real		Local
2	 int_1	0	Integer		Local
3	 bool_1	0	Boolean		Server
4	 str_1	12	String		Server

Fig. 7-3: Application Variables editor

With this editor, tag properties can be created, modified or deleted. Any tag and its properties can be used by means of the standard Windows commands cut (<Ctrl>+<X>), copy (<Ctrl>+<C>) or paste (<Ctrl>+<V>). It is also possible to undo the last modification to a field (<Ctrl>+<Z>).

Right-click and select the respective option from the context menu to sort data on the "Application Variables" sheet and/or insert/remove additional columns to/from the sheet.

Extending the Application Variables Datasheet

The Application Variables worksheet can be extended to up to 65,488 rows, if necessary.

The datasheet is normally limited to a maximum of 32,721 rows. (This is separate from the maximum size of the project database as a whole, as well as the runtime limit that is set when selecting a target platform for a new project.)

To extend the worksheet, edit the project file (**project_name.app**) to include the following entry:

[Options]

EnableExtendedTagCount=1

However, this brings the following restrictions:

- Project tags in rows 32,722-65,488 of the worksheet cannot be used as indices in expressions. I.e., an expression like **Abs(numArray[index-Tag]), indexTag** cannot be included in these lines. (This restriction does not apply to the VBScript interface.)
- In a Class worksheet, only the first 32 class members can have alarms. For any class members after the first 32, alarms do not work.

Working with Tags

Generally speaking, extending the "Application Variables" datasheet stretches the capabilities of "WinStudio" and should be done only if it is absolutely necessary.

7.1.3 Classes Folder

The "Classes" folder contains all of the project "Classes" and their respective class elements. Classes are compound structures consisting of the default data [Tag Types on page 113](#) ("Integer", "Real", "Boolean" and "String"). Classes allow for high-level data encapsulation in the project database. A class-type tag provides a set of values for its class elements.

To define a class, "class elements" and their types have to be defined. Class elements are variables with values for a project with particular features. Thus, defining a class can be very useful for projects with a repeatedly occurring group of variables.



Class variables are displayed in the **Application Variables** editor in the [Project Tags on page 106](#) folder with the class symbol .

To access the elements of a class, use the following syntax with a period (".") as separator: **TagName.MemberName**. For example: **tk.LEV** or **tk.TMP**.

If the "tk" tag is an [Array on page 114](#), the following syntax is to be used: **ArrayTagName[ArrayIndex].MemberName**

For example: **tk[1].LEV** or **tk[n].TMP**

A class tag is a set of values (instead of a single value) associated with the class. Class tags are created by grouping simple tags, which become the class elements. The maximum number of class elements for any class depends on the product specification. Class elements can include the standard types "Integer", "Real", "Boolean" and "String" as mentioned previously.

To create a new class, open the context menu of the "Class" folder and select the "Insert Class" entry.

When the "Insert Class" dialog is displayed, enter a class name in the **Name** field and click **OK** to close the dialog.

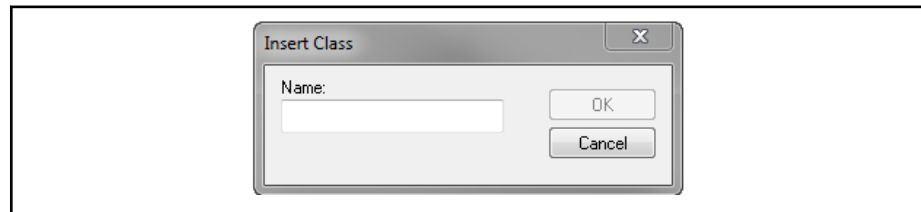


Fig. 7-4: *Insert Class dialog*



The class name has to be unique, i.e., it is not possible to create two classes with the same name.

"Distributed Variables" and "Internal Variables" cannot be configured as classes.

By opening the datasheet of a "Class" in the editor, class elements can be created, changed or deleted.

class_10			
	Name	Type	Description
1	 bool_02	Boolean	
2	 int_03	Integer	
3	 real_02	Real	
4	 str_02	String	

Fig. 7-5: Class datasheet



The maximum number of elements for each class is 512 as long as it does not exceed the maximum number of tags supported by the license (Product Type) selected for the project.

If a tag is created from a class type, each element from the class counts as one tag for licensing restrictions, because each class element has an independent value.

To edit a class element or property, right-click on the item and use the standard Windows commands to cut (<Ctrl>+<X>), copy (<Ctrl>+<C>), or paste (<Ctrl>+<V>). It is also possible to undo the last modification to a field (<Ctrl>+<Z>).

To edit class elements proceed as follows:

- **Name:** Enter a name for the class element or the element property. The first character has to be a letter and the name can contain up to 255 characters.
- **Type:** Select a data type ("Boolean", "Integer", "Real" or "String").
- **Description:** Enter a description of the class element for documentation purposes.



- Elements of a class cannot be of another class type.
- The class name has to be unique, i.e., it is not possible to re-use the name of an existing class.
- However: Elements with the same name can be created in different classes.

To delete a class and all its class elements, right-click on a class folder and select delete.

"WinStudio" disables the "Delete" option if any runtime tasks are executed.

Classes cannot be deleted as long as there is one variable of the class to be deleted.

7.1.4 Distributed Variables

The "Distributed Variables" folder contains all tags shared between "WinStudio" and the selected PC-based Control device. These tags have to be created and edited in the PC-based Control software and are then imported automatically into "WinStudio" under the following conditions:

- When starting "WinStudio" or
- When right-clicking on the "Shared Tags" folder to refresh (update) the database.

Working with Tags



Each PC-based Control software package has its own individual interface characteristics and conditions that allow "WinStudio" to import its tags. For example, the PC-based Control software application might have to be running to import database tags into "WinStudio".

If there are no PC-based Control software products associated with the application, the "Datasheet View" and "Tag List" subfolders (located in the "Distributed Variables" folder) are empty.

Distributed Variables cannot be edited in the "WinStudio" environment, however, they can be modified by means of the PC-based Control software.

However, Distributed Variables can be edited in any "WinStudio" task like any other tag.

Distributed Variables are read-only and viewable on the "Tag Properties" dialog and the "Distributed Variables" datasheet.



Right-click on the "Distributed Variables" folder (or click **Datasheet View**) and select the **Refresh** option to update the last "Version" of the PC-based Control software's tags database.

To edit the Shared Tags database (create a new tag, delete tags or edit tag properties) this command to update the "WinStudio" shared database has to be activated.

To view the "Tag Properties" dialog, proceed as follows:

- Click on the **Tag Properties** button in the "Cross References" tool window (the tag name has to be in the **Tag Name** field) or
- Double-click on the tag name in the [Project Tags on page 106](#) editor

If the "Distributed Variables" datasheet is displayed, it contains four columns ("Name", "Size", "Type" and "Description"). This datasheet is read-only and can only be used to view distributed variables.

7.1.5 Internal Variables

The "Internal Variables" folder contains predefined tags with specific functions (time, date, acknowledge alarms, storage of the logged user, etc.). These tags cannot be edited or deleted, however, their values can be accessed from any "WinStudio" task, copied and used elsewhere.



The internal variables database is updated by right-clicking on the "Internal Variables" element. For a list of the available internal variables including their properties and descriptions, refer to [List of System Tags on page 738](#).

The properties of internal variables can be displayed using the "Internal Variables" datasheet that contains four columns ("Name", "Size", "Type" and "Description").



Most internal variables are read-only.

For example, to change the time, the mathematical function [Set-SystemTime on page 524](#) has to be used instead of writing to the internal variable.

7.2 Creating Tags

7.2.1 Tag Name Syntax

When naming tags, the following guidelines have to be observed:

- Tag names **have to be unique**, i.e., same name cannot be specified for two different tags (or functions).
If an existing tag name is entered, "WinStudio" recognizes that the name already exists and does not create the new tag.
- Each tag name has to begin with a "letter". Additionally, letters, numbers and the underscore character ("_") can be used in tag names.
- The following symbols **cannot** be used tag names:
` ~ ! @ # \$ % ^ & * () - = / + \ [] { } < > ?
- Tag names or class elements may not exceed 255 characters.
Upper and lower case letters can be used. Tag names are **not** case sensitive.



As "WinStudio" does not differentiate between upper case and lower case characters, both can be used to make tag names more readable.

"TankLevel" instead of "tanklevel".

- Tag names have to differ from internal variables and math functions.



Use the "@" character at the beginning of a tag name to indicate that the tag is used in the project as an [Indirect Tag on page 119](#).

Some valid tag examples include:

- Temperature
- pressure1
- Count
- x

7.2.2 Choosing the Tag Type

"WinStudio" allows to create the following types of tags:

- **Basic tags** hold a single value
- **Array tags** are a set of tags that use the same name with unique indices
- **Class tags** are a set of compound tags that consist of user-defined data types (Boolean, Integer, Real or String) or data-type structures
- **Indirect tags** are pointers that provide indirect access to another tag type, including class tags

The following section describes these tag types in detail.

Basic tags

"Basic" tags receive single values. Typically, most tags defined for a project are basic tags.

Some examples of a basic tag include:

- "TankID" (to identify different tanks in the project)
- "Temperature" (to identify the current temperature of an object)
- "Status" (to identify whether an object is open or closed)

Working with Tags

Array tags An "array" tag consists of a set of tags that all have the same name but use unique indices (a matrix of "n" lines and one column) to differentiate between each tag. An "Array index" can be a fixed value, another tag or an expression. Maximum array sizes are determined by product specifications.

Array tags can be used for the following purposes:

- Simple configurations
- Enable multiplexing in screens, recipes and communication interfaces
- Saving development time during tag declaration

Array tags can be specified in two formats:

- For a simple array tag, enter:
ArrayTagName[ArrayIndex]
- For a complex array tag (where the array index is an expression consisting of a tag and an arithmetic operation), enter:

ArrayTagName[ArrayIndex + c]

The following apply here:

- **ArrayTagName** is the tag name
- **[ArrayIndex]** is the unique index (fixed value or another tag)
- "+" is an arithmetic operation; and
- **c** is a numerical constant



- For each array tag, a maximum index has to be specified by typing a value ("n") in the Array Size column of the "Application Variables" datasheet or in the Array Size field on a "New Tag" dialog (refer to [Application Variables on page 106](#)).

When creating an array tag with "n" positions, "WinStudio" actually creates "**n**+1" positions (from 0 to "n"). For example, if "ArrayTag[15]" is specified, the array has 16 elements, where 0 is the start position and 15 is the end position.

- Array tags may not contain any spaces.

If "WinStudio" reads a tag, it begins with the first character and continues until it finds the first space or zero. This way, the system does not recognize any characters following the space as part of the array tag.

For example, if **a[second + 1]** is entered, "WinStudio" recognizes **a[second** as the tag and considers it invalid because "WinStudio" does not find (recognize) the closing bracket. However, **a[second+1]** is recognized as a valid array tag.

Array tags can be specified wherever a variable name can be used. Also, as array tags greatly simplify configuration tasks and can save development time, it is recommended using them whenever possible.

As an example, temperature monitoring of four tanks is observed. The conventional configuration method is:

- Temperature1 – high temperature on tank 1
- Temperature2 – high temperature on tank 2
- Temperature3 – high temperature on tank 3
- Temperature4 – high temperature on tank 4

Working with Tags

Array tags can be used to simplify this task as follows (where **[n]** represents the tank number):

- Temperature[n] – high temperature on tank "n"

The following table provides some additional examples of array tags:

Array tag examples

Array tag examples	Description
Tank[1], Tank[2], Tank[500]	Simple arrays, where array indices ("1", "2" and "500") are numerical constants. For example, tank numbers.
Tank[tk]	A simple array, where the array index ("tk") is a tag. For example, a tag representing the tank number.
Tank[tk+1]	An array, where the index ("tk+1") is an expression. For example, the value of "tk" (tank number) plus 1.



If using another tag to reference the index of an array and if the value of the tag is outside the size of the array, the following results are given:

- If **IndexTag** is greater than the size of the array, **MyArray[IndexTag]** points to the end position of the array; and
- If **IndexTag** is less than 0, then **MyArray[IndexTag]** points to the start position of the array (i.e., **MyArray[0]**).

Indirect tags

Indirect tags "point" to other database tags (including class-type tags). Using indirect tags can save development time as no duplicate tags (and the logic built into them) have to be created.

Indirect tags can be created from any string-type tag simply by typing the "@" symbol in front of the tag name, i.e. **@TagName**.

- To reference a simple tag, it is assumed that the **strX** tag, (a String tag) contains the value "Tank", i.e. the name of a different tag. In this case, reading or writing to **@strX** provides access to the tag value **Tank**.
- To reference a class-type tag and member, simply create a string tag that points to the class tag and the member.

For example, if a tag **strX** (a string tag) holds the value "Tank.Level", which is the name of the class tag, then reading from or writing to **@strX** provides access to the tag value **Tank.Level**.

- It can also be pointed directly to a class-type tag member by identifying a class-type that points to a class member.

For example: to access the **Tank.Level** member of the class, the "Tank" value has to be stored within the **strX** tag and the **@strX.Level** syntax has to be used.

7.2.3 Choosing the Tag Data Type

Another consideration when designing a tag is what type of data the tag receives. "WinStudio" recognizes the following, standard tag "data types":

- **Boolean** ("one bit"): Simple Boolean value with the possible values of 0 (false) and 1 (true). Equivalent to the "bool" data type in C++. Typically used for turning objects off and on or for closing and opening objects.
- **Integer** ("four bytes"): Integer number (positive, negative or zero) internally stored as a signed 32-bit.

Working with Tags

Equivalent to the "signed long int" data type in C++. Typically used for counting whole numbers or setting whole number values.

Examples: **0, 5, -200.**

- **Real** ("floating point, eight bytes"): Real number that is stored internally as a signed 64-bit.

Equivalent to the "double" data type in C++. Typically used for measurements or for decimal or fractional values.

- **String** ("alphanumeric data, up to 1024 characters"): Character string up to 1024 characters that holds letters, numbers or special characters.

Supports both ASCII and UNICODE characters.

Examples: **Recipe product X123, 01/01/90, *** On ***.**

Tags can also be turned into a compound tag by assigning it a [Class on page 117](#).

A Class is a template consisting of two or more tag definitions, each with its own data type.

Classes can be used in projects that have items (e.g., tanks of liquid) with multiple attributes (e.g., fill level, temperature, pressure) to be monitored or controlled.

For further information, refer to: [Tag Properties and Parameters on page 124](#)

Changing how Boolean tags receive numeric values

When writing any numeric value not equal 0 (i.e., $\neq 0$) to a Boolean tag, the tag assumes a default value of 1.

If required, this behavior can be changed by editing the file **project_name.app** by changing the following settings:

[Options]

BooleanTrueAboveZero=value

If **BooleanTrueAboveZero** is set to the default value of "0", the project behaves as described above.

If **BooleanTrueAboveZero** is set to "1", the project behaves as follows:

- When writing any numeric value less than or equal to 0 (i.e., ≤ 0) to a Boolean tag, the tag assumes a value of 0 (false).
- When writing any numeric value greater than 0 (i.e., > 0) to a Boolean tag, the tag assumes a value of 1 (true).

NOTICE

This is a global runtime setting.

If it is only to be changed how certain tags are handled, this setting is not to be changed.

7.2.4 Using Array Tags

"WinStudio" tags can consist of a single value or an array of values.

The maximum array size is 16384 as long as it does not exceed the maximum number of tags supported by the license (Product Type) selected for the project.

Each array position (including the position 0) counts as one tag for licensing restrictions, because each position has an independent value.

An array tag is a set of tags with the same name that is identified by indices (a matrix of **n** lines and 1 column). The maximum array size depends on the product specification.

Working with Tags

The following syntax can be used to access an array tag:

`ArrayTagName[ArrayIndex]`

For example: **tank[0]**, **tank[1]**, **tank[2]** and **tank[500]**.

A maximum index for each array tag has to be specified in the **size** column of each datasheet.

n can be specified to indicate the array tag has positions from 0 to **n**.

For example, if the size of TagA is 3, the tag elements could be **TagA[0]**, **TagA[1]**, **TagA[2]** and **TagA[3]**.



Use the array tag whenever possible as it optimizes memory use and simplifies the configuration task.

For example, if a display is required to monitor each tank, array tags can be used to configure a single display containing tags linked to all tanks.

For example (using the **tk** tag as an index containing the number of the tank): **pressure[tk]**, **temperature[tk]** and **temperature[tk+1]**.

An array tag can be a tag, a numeric value or an expression with the arithmetic operator "+".

When referring to an array with an index using the "+" arithmetic operation, the following syntax has to be used:

`ArrayTagName[NumValue1+NumValue2]`

Where **NumValue1** and **NumValue2** can be an integer tag or a numerical constant.

For example: **temperature[tk+2]** or **temperature[tk+6]**.

Using array tags in any "WinStudio" task can save a significant amount of project development time.

For example, if tag points related to the temperature of four tanks are required. The conventional configuration method is the following:

- **Temperature1**: high temperature on tank 1
- **Temperature2**: high temperature on tank 2
- **Temperature3**: high temperature on tank 3
- **Temperature4**: high temperature on tank 4

Using array tags simplifies this task, as follows:

- **Temperature[j]**: high temperature on tank {j}

If a four-position array tag is created, the system creates five positions (from 0 to 4).

For example:

```
tag_example[15] //start position=0, end position=15
```

Therefore, the **tag_example[15]** array has 16 elements.

If using another tag to reference the index of an array and if the value of the tag is outside the limits of the array, the following results are given:

- If **IndexTag** is greater than the size of the array, **MyArray[IndexTag]** points to the end position of the array; and
- If **IndexTag** is less than 0, **MyArray[IndexTag]** points to the start position of the array.

Working with Tags

Array tags An "array" tag consists of a set of tags that all have the same name but use unique indices (a matrix of **n** lines and one column) to differentiate between each tag. An "Array index" can be a fixed value, another tag or an expression. Maximum array sizes are determined by product specifications.

Array tags can be used for the following purposes:

- Simple configurations
- Enable multiplexing in screens, recipes and communication interfaces
- Saving development time during tag declaration

Array tags can be specified in one of two formats:

- For a simple array tag, enter:
ArrayTagName[ArrayIndex]
- For a complex array tag (where the array index is an expression consisting of a tag and an arithmetic operation), enter:

ArrayTagName[ArrayIndex + c]

The following apply here:

- **ArrayTagName** is the tag name
- **[ArrayIndex]** is the unique index (fixed value or another tag)
- "+" is an arithmetic operation; and
- **c** is a numerical constant
- For each array tag, a maximum index has to be specified by entering a value (**n**) in the Array Size column of the "Application Variables" data-sheet or in the Array Size field on a "New Tag" dialog.

(See ["Creating project database Tags on page 106"](#)).

When creating an Array Tag with **n** positions, "WinStudio" actually creates **n+1** positions (from **0** to **n**).

For example, if **ArrayTag[15]** is specified, the array has 16 elements, where 0 is the start position and 15 is the end position.

- Array tags may not contain any spaces. If "WinStudio" reads a tag, it begins with the first character and continues until it finds the first space or zero. Consequently, the system does not recognize any characters following the space as part of the array tag.

For example, if **a[second + 1]** is entered, "WinStudio" recognizes **a[second** as the tag and considers it invalid because "WinStudio" does not find (recognize) the closing bracket.

However, if **a[second+1]** is entered, it is recognized as a valid array tag

Array tags can be specified everywhere, where a variable name would be used. As array tags greatly simplify configuration tasks and can save development time, it is recommended using them whenever possible.

As an example, temperature monitoring of four tanks is observed. The conventional configuration method is:

- **temperature1** – high temperature on tank 1
- **temperature2** – high temperature on tank 2
- **temperature3** – high temperature on tank 3
- **temperature4** – high temperature on tank 4

Array tags can be used to simplify this application task as follows (where **[n]** represents the tank number):

- **Temperature[n]** – high temperature on tank **[n]**

The following table provides some additional examples of array tags:

Array tag examples

Array tag examples	Description
Tank[1], Tank[2], Tank[500]	Simple arrays, where the array indexes (1, 2, and 500) are numerical constants. For example, tank numbers.
Tank[tk]	A simple array, where the array index (tk) is a tag, e.g. a tag representing a tank number.
Tank[tk+1]	Compound, at which the Array index (tk+1) is an expression, e.g., the value of tk (tank number) plus 1.

If using another tag to reference the index of an array and if the value of the tag is outside the limits of the array, the following results are given:

- If **IndexTag** is greater than the size of the array, **MyArray[IndexTag]** points to the end position of the array; and
- If **IndexTag** is less than 0, then **MyArray[IndexTag]** points to the start position of the array (i.e., **MyArray[0]**).

7.2.5 Using Classes

"Class" tags are compound tags that permit a high-degree of encapsulation within the Tags database. Where basic tags receive a single value, "classes" are designed to receive multiple values.

Class tags can be created by grouping basic or array tags, which then become class "members". The maximum number of members for any class depends on the product specification.

Class tags can be specified in two formats:

- For a simple class tag the syntax is
 TagName.ClassMemberName
 (Where the period is used as a separator.)
 For example, several different conditions are to be monitored (such as "temperature", "level" and "pressure") in a tank, a class tag can be created as follows:
 - Tank.Temperature
 - Tank.Level
 - Tank.Pressure
- For creating a compound class tag (using an array tag) the syntax is
 ArrayTagName[ArrayIndex].ClassMemberName
 (again, the period is used as a separator.)
 If "temperature", "level" and "pressure" conditions are to be monitored in multiple tanks, a class tag can be created as follows:
 - Tank[tk].Temperature
 - Tank[tk].Level
 - Tank[tk].Pressure

Where "tk" is an array index representing the tank number.

Classes folder

The **Classes** folder includes all project classes.

Classes are compound tags consisting of user-defined data-type structures or [Tag Types on page 113](#) ("Integer", "Real", "Boolean" and "String").

Working with Tags



Classes allow for high-level data encapsulation in the project database.

A class-type tag provides a set of values for its class elements.

To define a class, "class elements" and their types have to be defined.

Class elements are variables that hold values for an object with particular characteristics. Thus, defining a class can be very useful for projects with a repeatedly occurring group of variables.

Classes in the "Classes" folder as well as tags of the "Class" data type in the [folder, Application Variables on page 106](#) are indicated by the symbol .

To access the elements of a class, use the following syntax with a period (".") as separator:

TagName.MemberName

For example: **tk.LEV** or **tk.TMP**.

If the "tk" tag is an [Array on page 114](#), the following syntax is to be used:

Program:

ArrayTagName[ArrayIndex].MemberName

For example: **tk[1].LEV** or **tk[n].TMP**

A class-type tag contains a set of values (rather than a single value) associated with the class.

Class tags are created by grouping simple tags, which become the class elements. The maximum number of elements for any class depends on the product specification. Class elements can include the standard values "Integer", "Real", "Boolean" and "String" as mentioned previously.

To create a new class, use one of the following methods to open the "Insert Class" dialog:

- Select the "New Class" entry in the context menu of the "classes" folder; or
- Create a new class tag in the [Project Tags on page 106](#) folder.

If the "Insert Class" dialog displays, enter a class name in the **Name** field and then click **OK** to close the dialog.

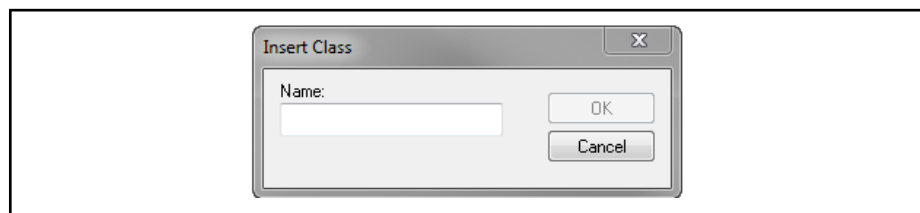
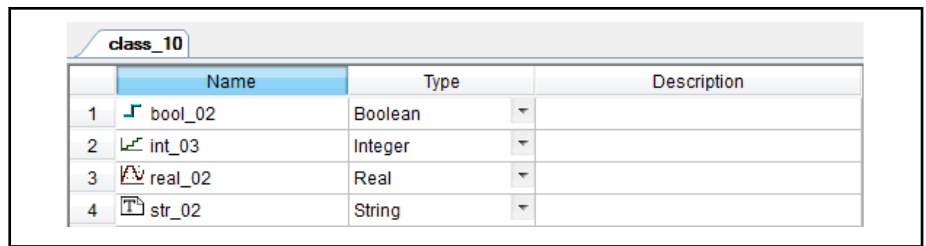


Fig. 7-6: Insert Class dialog

A unique class name has to be entered. It is not possible to create two classes with the same name. Additionally, shared tags and system tags cannot be configured as classes.

If the "Class" datasheet is displayed, it can be used to create, modify or delete any class elements and their viewable properties. (Classes cannot be edited from the "Tag Properties" dialog.)



	Name	Type	Description
1	bool_02	Boolean	
2	int_03	Integer	
3	real_02	Real	
4	str_02	String	

Fig. 7-7: Class datasheet

The maximum number of class elements for each class is 512 as long as it does not exceed the maximum number of tags supported by the license (Product Type) selected for the project. If a tag is created from a class type, each element from the class counts as one tag for licensing restrictions, because each class element has an independent value.

To edit a class element or property, right-click on the item and use the standard Windows commands to cut (<Ctrl>+<X>), copy (<Ctrl>+<C>) or paste (<Ctrl>+<V>). The last modification to a field can also be undone by <Ctrl>+<Z>.

To edit element properties, proceed as follows:

- **Name** field: Enter a name for the class element or the element property. The first character has to be a letter and the name can contain up to 255 characters
- **Type** combo box: Select a data type ("Boolean", "Integer", "Real" or "String").
- **Description** field: Enter a description of the class element for documentation purposes

Elements of a class cannot be of another class type. Also, a unique class name has to be created. The name of an existing class cannot be reused. However, elements with the same name can be created in different classes.

To delete a class and all its elements, right-click on a class folder and select delete.

"WinStudio" disables the delete option if any runtime tasks are running.

In addition, classes cannot be deleted if they are associated with any tag.

7.2.6 Using Indirect Tags (Pointers)

"WinStudio" supports indirect access to tags in the database.

Example: a tag **X** of the String type.

This tag can hold the name of any other tag in the database (i.e., it can provide a pointer to any other type of tag, including a class type).

Syntax for an indirect tag:

@IndirectTagName

A tag named **X** holds a string "TEMP".

Reading and/or writing to **@X** provides access to the value of the "TEMP" variable.

Any tag created as a string-type tag is potentially an indirect tag (pointer).

To refer to a class-type tag, a String-type tag can be referred to that points to a class tag.

Working with Tags

Class	"TANK" with element "Level"
Tag	"TK" of the class "TANK"
Tag	"XCLASS" of the String type

To access the **TK.Level** value, the **TK.Level** value has to be stored within the "XCLASS" tag and the **@XCLASS** syntax has to be used.

It is also possible to refer directly to an element of a class-type tag, identifying a class-type that points to a class element.

For example:

Class	"TANK" with element "Level"
Tag	"TK" of the class "TANK"
Tag	"XCLASS" of the class "TANK"

To access the **TK.Level** value, the **""TK""** value has to be stored within the "XCLASS" tag and the **@XCLASS.Level** syntax has to be used.

When creating tags for indirect use, place an **X** in the tag column rather than creating them as Strings. For type, enter the type of tag a reference is created for.

Follow the "XCLASS" example: **@Z Integer**, **@X Class:TANK**.

Indirect tags

Indirect tags "point" to other database tags.



Using indirect tags can save development time as no duplicate tags (and the logic built into them) have to be created.

Indirect tags can be created from any String-type tag simply by typing the "@" symbol in front of the tag name, i.e. **@TagName**.

- To reference a simple tag, assume the "strX" tag (a String tag) holds the value "Tank", which is the name of another tag, then reading from or writing to **@strX** provides access to the value of the "Tank" tag.
- To reference a class-type tag and element, simply create a String tag that points to the class tag and the element.

For example, if a tag "strX" (a string tag) holds the value **""Tank.Level""**, which is the name of the class tag, then reading from or writing to **@strX** provides access to the value of the **Tank.Level** member.

- It is also possible to refer directly to an element of a class-type tag, identifying a class-type that points to a class element.

For example: to access the **Tank.Level** element of the class, the "Tank" value has to be stored within the "strX" tag and the **@strX.Level** syntax has to be used.

7.2.7 Choosing the Tag Scope

"WinStudio" allows to decide whether a tag creates only one instance on a project server or copies on each local station.

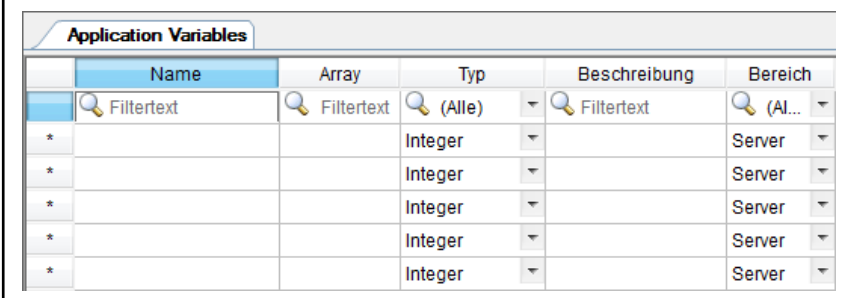
- **Server** (default): The tag is created on the project server and shared by all connected clients (e.g., Thin Client, Secure Viewer). A change to the tag value affects the entire project.
- **Local**: A virtual copy of the tag is maintained separately on each local station (server + clients), a change to the tag value affects only the station on which the change was made.

7.3 Creating Database Tags

7.3.1 Creating Tags on the Datasheet

To create tags on the "Application Variables" datasheet, proceed as follows:

1. Open the **Variables** folder
2. Open the "Application Variables" datasheet by double-clicking on the respective element in the Project Explorer:



Application Variables				
Name	Array	Typ	Beschreibung	Bereich
Filtertext	Filtertext	(Alle)	Filtertext	(Al...
*		Integer		Server
*		Integer		Server
*		Integer		Server
*		Integer		Server
*		Integer		Server

Fig.7-8: Application Variables datasheet

3. Locate an empty line in the datasheet and configure the following fields.

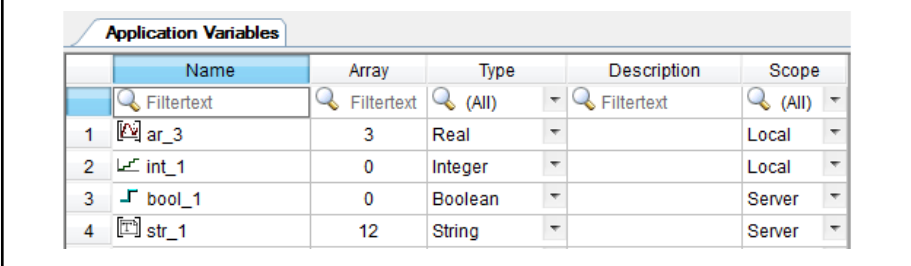


The Tab key on the keyboard can be used to move to the next column.

- **Name** column: Enter a name using the proper syntax.
(For further information, refer to [Tag Name Syntax on page 111](#))
 - **Array size** column:
 - For an array tag, enter a value to specify the maximum index of the array.
 - For any other tag type, enter zero (0).
 - **Type** combo box: Click the arrow to select a tag data type (Boolean, Integer, Real, or String) from the list.
(For further information, refer to [Choosing a Tag Data Type on page 113](#))
 - **Description** column (optional): Enter a description for documentation purposes only.
 - **Scope** combo box: Click the arrow to specify whether the tag value is shared with (displayed on) Thin Client stations.
(For further information, refer to [Choosing the Tag Scope on page 120.](#))
4. Click into a new line to create further tags.
If no more tags are to be created, close the "Application Variables" datasheet.

The following example illustrates a variety of tags configured in an "Application Variables" datasheet.

Working with Tags



Application Variables				
	Name	Array	Type	Scope
	Filtertext	Filtertext	(All)	(All)
1	ar_3	3	Real	Local
2	int_1	0	Integer	Local
3	bool_1	0	Boolean	Server
4	str_1	12	String	Server

Fig. 7-9: Example for datasheet Application Variables

7.3.2 Creating Tags "On-the-Fly"

Instead of opening the Application Variables datasheet, every time a new tag is to be created, individual tags can be created "on-the-fly" by performing any of the following actions:

- In the "Project Explorer", in the context menu of the "Application Variables" element, select the "Insert Variables" entry.
- Enter a new tag name into any **Tag/Expression** text field (available from "Object Properties" dialog, worksheets, etc.). If the "Question" dialog asks whether a new tag is to be created, click **Yes**.

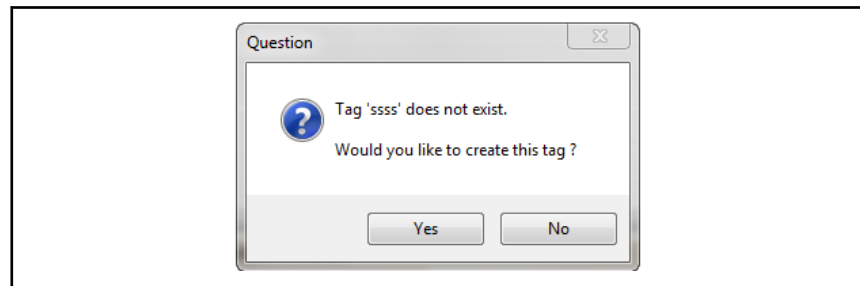


Fig. 7-10: Creating a new tag

Any of these actions causes a "New Tag" dialog to be displayed that can then be completed as needed.

(For further information, refer to [Application Variables, General Information on page 106.](#))

7.3.3 Editing Tags

Changes in the Application Variables datasheet

The properties of a tag can be changed at any time during development or runtime.

Use the following steps to edit tags in the "Application Variables" datasheet:

1. Open the "Application Variables" datasheet and look for the desired tag.
2. Double-click in the column containing the information to be changed, and enter the new information into the datasheet.
3. After editing, close the datasheet. Changes are applied immediately into the tags database.



- 3.1 Right-click and select the respective option from the context menu to sort data on the "Application Variables" sheet and/or insert/remove additional columns to/from the sheet
- 3.2 Any tag and its properties can be used by means of the standard Windows commands cut (<Ctrl>+<X>), copy (<Ctrl>+<C>) or paste (<Ctrl>+<V>)
- 3.3 It is also possible to undo the last modification to a field (<Ctrl>+<Z>)

7.4 Creating Classes

To create a new class tag, proceed as follows:

1. Open the context menu of the "Classes" element and select the **Insert Class** entry
2. If the "Insert Class" dialog is displayed, enter a **Name** into the respective text field

(For details, refer to [Tag Name Syntax on page 111](#) and [Choosing the Tag Type on page 111](#))

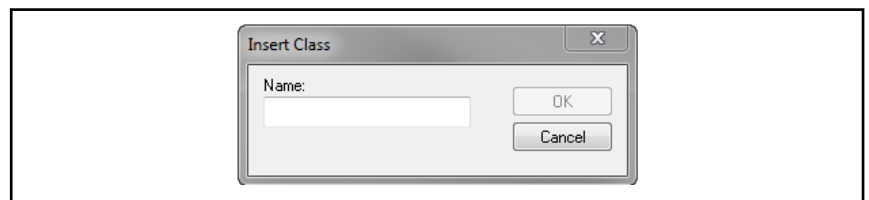


Fig.7-11: Insert Class dialog

3. Click **OK** to close the "Insert Class" dialog. A new element is created in the Project Explorer. Open this element by double-click.

The columns in this worksheet are configured as follows:

- **Name:** Enter a class element name.
- **Type** selection list: Click the arrow to select a data type (Boolean, Integer, Real, or String) for the class element from the list.
- **Description** (optional): Enter a description of the class element for documentation purposes.

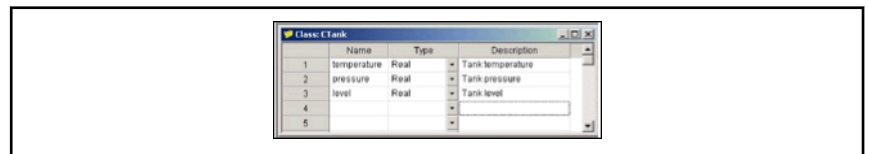


Fig.7-12: Example for the "Class" worksheet

4. Click into the next blank line and enter the information for the next class element to be included in this class. If no more elements are to be added, the "Class" worksheet can be closed.
5. Afterwards, refer to the instructions under [Creating Tags on the Data-sheet on page 121](#) to create and connect a tag to a new class.

Please note that the new class name ("CTank") is included in the types list if the arrow button for display is pressed (refer to line 5 in the following figure). Select the class name from this list.

Working with Tags

Application Variables					
	Name	Array	Type	Description	Scope
	Filtertext	Filtertext	(All)	Filtertext	(All)
1	Valve_Fill_State	3	Boolean	Fill valve state (open/close)	Server
2	Valve_Empty_State	3	Boolean	Empty valve state (open/close)	Server
3	Valve_Fill_Command	3	Boolean	Fill valve command	Server
4	Valve_Empty_Comm...	3	Boolean	Empty valve command	Server
5	Tank	3	CTank	Tank Data	Server

Fig.7-13: Creating the Tank class tag

6. After finishing, save the work and close the worksheet.

7.5 Setting Tag Properties

7.5.1 Tag Properties and Parameters

Tag types have the following properties:

- **Parameters**
- **Alarms properties**
- **History properties**

To configure the properties of a tag, open the "Cross Reference" tool window, enter the tag name into the text field and click on the **Tag Properties** button.

The "Tag Properties" dialog is displayed:

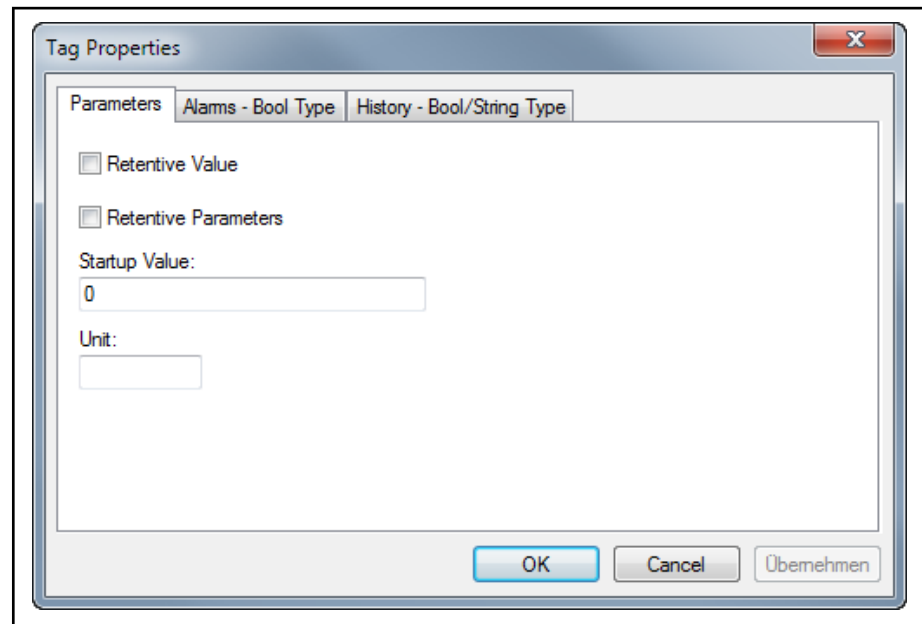


Fig.7-14: Tag Properties dialog for a Boolean tag

Use the parameters in this dialog to configure the different tag properties.

Please refer to the instructions below for type-related configuration:

String-type tag parameters

To specify String-type parameter properties from the "Tag Properties" dialog, proceed as follows:

- **Retentive Value:** Activate this checkbox to permanently save the tag value in case of an unexpected system crash. The system can be restarted from the last saved value.

Working with Tags

- **Retentive Parameters:** Click (check) to save runtime changes in the tag's properties. Only specific properties can be saved.

For a complete list, please refer to [List of Tag Properties on page 129](#).

NOTICE

Enabling either of the retentive options for tags that frequently change values can cause heavy disk access, which slows performance.

- **Startup Value:** Enter a tag value for the application start.
The tag assumes this value if the **Retentive Value** option is disabled.
- **Unit:** Enter any string (up to 9 characters) as a brief description or reference.

This tag is accessible at runtime.



"WinStudio" does not accept writing values that fall outside the arrays defined in the **Min** and **Max** fields.

In addition, "WinStudio" generates a message in [LogWin Output Window on page 85](#) to indicate that the system tried to write a value outside the defined array

Integer and Real-type tag parameters



For a description of **Retentive Value**, **Retentive Parameters**, **Startup Value** and **Unit** parameter, refer to ["String-type tag parameters" on page 124](#)

To specify Integer and Real-type parameters in the "Tag Properties" dialog, proceed as follows:

- *Units range*
 - **Min:** Enter a minimum value for the tag in engineering units. This tag is accessible at runtime.
 - **Max:** Enter maximum value for the tag in engineering units. This tag is accessible at runtime.
 - **Unit:** Enter any string (up to 9 characters) as a brief description or reference of the tag. This tag is accessible at runtime.
- *Signal condition range*
 - **Insensitive range (dead band):** Enable (check) to insert the dead band value of a tag.
The dead band value is a variation around a central value of the tag, that does not trigger an alarm.
 - **Smoothing:** Enable (click) to save the rate of change for the tag properties.

Use only for Integer and Real tags.

For example, if the **Smoothing** option has been selected for the **LEVEL1** tag containing the value = 50, during the next search, if the **LEVEL1** changes to 60, the system stores the average of 50 + 60 in the database, i.e. the new value is set to = 55.

Boolean-type tag parameters



For a description of **Retentive Value**, **Retentive Parameters**, **Startup Value** and **Unit** parameter, refer to ["String-type tag parameters" on page 124](#)

Working with Tags

Also refer to:

- [Using Tag Properties: Alarms on page 126](#)
- [Using Tag Properties: History on page 129](#)

7.5.2 Using Tag Properties: Alarms

Use the [Tag Properties on page 140](#) dialog to view the configured alarms for a selected tag. "WinStudio" disables this command if there are open alarm worksheets.

Before using these dialogs, alarm groups have to already have been created.

ALARM TYPES:

- **HiHi:** A very high alarm is active
- **Hi:** A high alarm is active
- **Lo:** A low alarm is active
- **LoLo:** A very low alarm is active
- **Rate:** An alarm based on the rate of change is active
- **Deviation:** An alarm based on deviation from a given set point is active

Example:

If SetPoint = 50, Deviation + = 5, Deviation - = 5 and Deviation Dead Band = 0.5;

"WinStudio" generates an alarm when the temp ≥ 55 or temp ≤ 45 ; and it is returned to normal state if temp ≤ 54.5 or temp ≥ 45.5 .

ALARM LIMITS:

- **HiHiLimit:** When creating very high alarms in the "Tag Properties" dialog, this field is used to specify the limits.
This field can be accessed at runtime and used for modifications on-the-fly.
- **HiLimit:** When creating high alarms in the "Tag Properties" dialog, this field is used to specify the limits.
This field can be accessed at runtime and used for modifications on-the-fly.
- **LoLimit:** When creating low alarms in the "Tag Properties" dialog, this field is used to specify the limits.
This field can be accessed at runtime and used for modifications on-the-fly.
- **LoLoLimit:** When creating Very Low alarms in the "Tag Properties" dialog, this field is used to specify the limits.
This field can be accessed at runtime and used for modifications on-the-fly.
- **DevSetpoint:** Reference point for a tag value deviation that triggers an alarm. The alarm message is to be defined in the "Tag Properties" dialog or on an "Alarm" worksheet.
This field can be accessed at runtime.
- **Dev+Limit:** Limit deviation to a value higher than the **DevSetpoint** in a tag property that triggers an alarm. The alarm message is to be defined in the "Tag Properties" dialog or on an "Alarm" worksheet.
This field can be accessed at runtime.

Working with Tags

- **Dev+Limit:** Limit deviation to a value lower than the **DevSetpoint** in a tag property that triggers an alarm. The alarm message is to be defined in the "Tag Properties" dialog or on an "Alarm" worksheet.

This field can be accessed at runtime.

- **RateLimit:** Limit of rate variation in a tag value that triggers an alarm. The alarm message is to be defined in the "Tag Properties" dialog or on an "Alarm" worksheet.

This field can be accessed at runtime.

Alarms for Integer and Real-type tags

To specify an alarm for Integer and Real-type tags in the "Tag Properties" dialog, proceed as follows:

- **Alarms enabled:** Click (activate) this checkbox to enable according to configuration
- **Remote Ack tag:** Enter a tag to acknowledge this alarm
- **Dead zone:** Enter a value for the filter for alarms normalization.

Example:

If the TEMP1 tag is configured with an alarm with the Hi limit = 90 and dead zone = 5,

"WinStudio" generates an alarm if $TEMP1 \geq 90$ and returns to normal state if $TEMP1 \leq 85$.

- **Translation Enabled:** Click (check) to enable the translation of messages defined on the "Options" tab in the [Project Settings on page 62](#) dialog.

For additional information on translations, refer to [Translation Tools on page 381](#).



"WinStudio" saves alarm messages with the **Translation Enabled** attribute enabled in a file called "Alarm.TXT" located in the project "DATABASE\\" directory.

- **HiHi (HiHiLimit):** Click (check) to indicate a very high alarm is active. Accessible at runtime.
- **Hi (HiLimit):** Click (check) to indicate a high alarm is active. Accessible at runtime.
- **Lo (LoLimit):** Click (check) to indicate a low alarm is active. Accessible at runtime.
- **LoLo (LoLoLimit):** Click (check) to indicate a very low alarm is active. Accessible at runtime.
- **Rate (RateLimit):** Click (check) to indicate a rate alarm is active. Accessible at runtime.
- **Deviation+:** Click (check) to indicate a deviation alarm is active. Accessible at runtime.
- **Deviation-:** Click (check) to indicate a deviation alarm is active. Accessible at runtime.
- **Deviation SetPoint:** Enter a reference point for the deviation. Accessible at runtime.
- **Deviation dead zone:** Enter a reference value for the deviation.

Working with Tags

Alarms for Boolean-type tags

Use the "Alarms - Bool Type" tab on the "Tag Properties" dialog to specify alarm properties for Boolean-type tags.

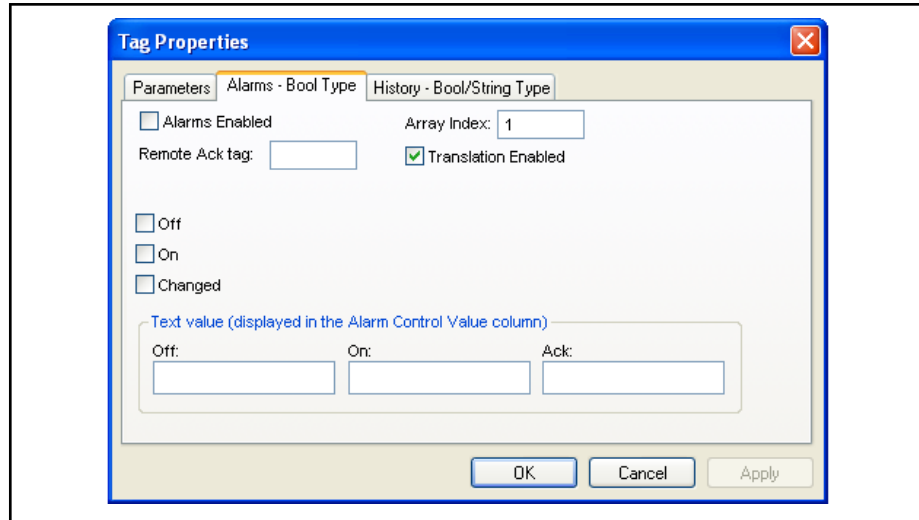


Fig. 7-15: Alarms - Bool Type tab

Configure the parameters as follows:

- **Alarms Enabled:** Click (check) to enable according to configuration.
- **Remote Ack tag:** Enter a tag to enable remote alarm acknowledgement, which occurs if the tag values change.
- **Translation Enabled:** Click (check) to enable the translation of messages defined on the "Options" tab in the [Project Settings on page 62](#) dialog.

For additional information on translations, refer to [Translation Tools on page 381](#).



"WinStudio" saves Alarm messages with the **Translation Enabled** attribute enabled in a file called **Alarm.TXT** located in the project **\DATABASE** directory.

- **Off:** Click (enable) to always generate an alarm message if the tag value is zero.
- **On:** Click (enable) to always generate an alarm message if the tag value is one.
- **Changed:** Click (enable) to always generate an alarm message if the tag value changes.
- **Text value:** Use the **Off**, **On**, and **Ack** fields to configure mnemonics (for example, "Closed" or "Open") for the **Off**, **On**, and **Ack** alarm states (Boolean tags only).

At runtime, the Alarm Control object displays this text in the "Value" column of the alarm message.

This text can also be accessed using the [Tag Fields on page 129](#).



If no mnemonics are configured, the alarm control object displays the tag value ("0" or "1") in the "Value" column.

7.5.3 Using Tag Properties: History

Use the [Tag Properties on page 140](#) dialog to view the history for a selected tag. "WinStudio" disables this command if there are open trend worksheets. Before using this dialog, the trend groups have to already have been created.



History does not support String-type tags.

Refer to [Recipes on page 319](#) folder for further information about storing String values.

Integer and Real-type history

In the "Tag Properties" dialog, enable history for Integer and Real-type tags as follows:

- **Enable history:** Click (check) to enable storage of the selected tag value samples.
- **Group number:** Specify the group number to which this tag is associated.
- **Log dead band:** Specify a value sample taken when the variation value is equal to, or greater than, the **Log Dead Band**.

Boolean-type history

To enable the history for Boolean-type tags in the "Tag Properties", proceed as follows:

- **Enable history:** Click (check) to enable storage of the selected tag value samples.
- **Group number:** Specify the group number to which this tag is associated.

7.5.4 List of Tag Properties

Tag properties (also known as "tag fields") are metadata attached to each tag in the database. Most of these properties can be set using the "Tag Properties" dialog that can be opened by clicking the **Tag Properties** button in the "Cross References" tool window.

To access a tag property at runtime, use the following syntax (without spaces) where tags would usually be specified:

tag_name->property_name

The following tag properties can be accessed at runtime:

Tag property	Description	R or R/W	Data type	Available on data type...				Retentive
				Bool	Int	Real	Str	
Name	The name of the tag, as configured in the Project Tags database.	R	String, up to 32 chars	Y	Y	Y	Y	n/a
Member-Name	The name of the class element, in a properly configured class on page 108 . NOTE: The syntax has to be: Class.Member->MemberName Example: Tank.Lvl->MemberName = Lvl	R	String, up to 32 chars	Y	Y	Y	Y	n/a
Size	Array size, if the tag is not an array tag, it returns the value 0	R	Integer	Y	Y	Y	Y	n/a

Working with Tags

Tag property	Description	R or R/W	Data type	Available on data type...				Retentive
				Bool	Int	Real	Str	
Index	The index number of an element in an array on page 114. (An array is any tag larger than 0.) NOTICE:The syntax has to be: Tag[Index]->Index Example: Tag[1]->Index = 1	R	Integer	Y	Y	Y	Y	n/a
Description	The description of the tag, configured in the tags datasheet.	R	String	Y	Y	Y	Y	Y
Quality	Tag quality (192=GOOD; 0=BAD). The project updates this field every time the tag receives the result of an expression or a value from a communication task (e.g. as driver or OPC). If the expression is invalid (e.g. division by zero) or if there is a reading communication error associated with the tag, the project sets the quality to BAD.	R	Integer	Y	Y	Y	Y	N
TimeStamp	Time and date of the last change of the tag value.	R	String	Y	Y	Y	Y	N
Blocked	This property can have two values: This is useful when all actions associated with a specific tag are to be dynamically deactivated. Even if a tag is blocked, it still counts towards the total number of tags for licensing purposes. 0: The tag is blocked and all runtime tasks ignore it. It is effectively removed from the project database. 1: The tag is unblocked and all runtime tasks can access it normally.	R/W	Boolean	Y	Y	Y	Y	N
Unit	A brief description (up to 9 characters) of the engineering unit (i.e., the unit of measurement) for the tag value. For example, Kg, kWh, Pa .	R/W	String, up to 9 chars	Y	Y	Y	Y	Y
Max	The maximum value that can be written to the tag at runtime.	R/W	Real	N	Y	Y	N	Y
Min	The minimum value that can be written to the tag at runtime	R/W	Real	N	Y	Y	N	Y
B0 ... B31	Value (0 or 1) of any of the 32 bits (b0, b1, b2, ... b31) of an Integer tag. (B0: LSB B31: MSB)	R/W	Boolean	N	Y	N	N	N

Working with Tags

Tag property	Description	R or R/W	Data type	Available on data type...				Retentive
				Bool	Int	Real	Str	
DisplayValue	A converted tag value that is only displayed on-screen. DisplayValue = (Value / UnitDiv) + UnitAdd This is used when the actual tag values have one engineering unit (see Unit above) but need to be displayed on-screen in another engineering unit (see DisplayUnit below). For example, degrees Celsius and Fahrenheit. If the user changes DisplayValue at runtime, the conversion is reversed before the change is actually written to the tag: Value = (DisplayValue – UnitAdd) * UnitDiv	R/W	Real	N	Y	Y	N	n/a
DisplayUnit	A brief description (up to 9 characters) of the engineering unit for DisplayValue. NOTE: This property can only be set by using the SetDisplayUnit on page 605 and SetTag-DisplayUnit on page 606 functions.	R	String, up to 9 chars	N	Y	Y	N	N
UnitDiv	Number by which the tag value is divided to get DisplayValue. To perform no division, UnitDiv is to be 1. NOTE: This property can only be set by using the SetDisplayUnit on page 605 and SetTag-DisplayUnit on page 606 functions.	R	Real	N	Y	Y	N	N
UnitAdd	Number added to the tag value to get DisplayValue. To perform no addition, UnitAdd is to be 0. NOTE: This property can only be set by using the SetDisplayUnit on page 605 and SetTag-DisplayUnit on page 606 functions.	R	Real	N	Y	Y	N	N
DisplayMax	The maximum value that can be entered to DisplayValue at runtime. DisplayMax = (Max / UnitDiv) + UnitAdd If DisplayMax is changed at runtime, Max is also changed as follows: Max = (DisplayMax – UnitAdd) * UnitDiv	R/W	Real	N	Y	Y	N	N
DisplayMin	The minimum value that can be entered to DisplayValue at runtime. DisplayMin = (Min / UnitDiv) + UnitAdd If DisplayMin is changed at runtime, Min is also changed as follows: Min = (DisplayMin – UnitAdd) * UnitDiv	R/W	Real	N	Y	Y	N	N
HiHiLimit	Limit value for the HiHi alarm	R/W	Real	N	Y	Y	N	Y
HiLimit	Limit value for the Hi alarm	R/W	Real	N	Y	Y	N	Y
LoLimit	Limit value for the Lo alarm	R/W	Real	N	Y	Y	N	Y

Working with Tags

Tag property	Description	R or R/W	Data type	Available on data type...				Retentive
				Bool	Int	Real	Str	
LoLoLimit	Limit value for the LoLo alarm	R/W	Real	N	Y	Y	N	Y
RateLimit	Limit value for the Rate alarm	R/W	Real	N	Y	Y	N	Y
DevSetpoint	Setpoint value for Deviation alarms	R/W	Real	N	Y	Y	N	n/a
DevPLimit	Limit value for the Deviation+ alarm	R/W	Real	N	Y	Y	N	Y
DevMLimit	Limit value for the Deviation- alarm	R/W	Real	N	Y	Y	N	Y
HiHi	If 0, the HiHi alarm is not active. If 1, the HiHi alarm is active.	R	Boolean	Y	Y	Y	N	n/a
Hi	If 0, the Hi alarm is not active. If 1, the Hi alarm is active.	R	Boolean	Y	Y	Y	N	n/a
Lo	If 0, the Lo alarm is not active. If 1, the Lo alarm is active.	R	Boolean	Y	Y	Y	N	n/a
LoLo	If 0, the LoLo alarm is not active. If 1, the LoLo alarm is active.	R	Boolean	Y	Y	Y	N	n/a
Rate	If 0, the Rate alarm is not active. If 1, the Rate alarm is active.	R	Boolean	Y	Y	Y	N	n/a
DevP	If 0, the Deviation+ alarm is not active. If 1, the DevP alarm is active.	R	Boolean	N	Y	Y	N	n/a
DevM	If 0, the Deviation- alarm is not active. If 1, the DevM alarm is active.	R	Boolean	N	Y	Y	N	n/a
AlrStatus	Integer value with the status of the current active alarm associated with the tag. Each bit of this integer value indicates a specific status: • Bit 0 (LSB): HiHi Alarm active • Bit 1: Hi Alarm active • Bit 2: Lo Alarm active • Bit 3: LoLo Alarm active • Bit 4: Rate Alarm active • Bit 5: Deviation+ Alarm active • Bit 6: Deviation- Alarm active Examples: If Tag->AlrStatus returns the value 2, it means that the Hi alarm is active. If it returns the value 3, it means that the HiHi and the Hi alarms are active simultaneously. If this property returns the value 0, it means that there are no active alarms associated with this tag. For Boolean tags, only the values 1 (bit 1), 4 (bit 2) or 16 (bit 4) can be returned.	R	Integer	Y	Y	Y	N	N

Working with Tags

Tag property	Description	R or R/W	Data type	Available on data type...				Retentive
				Bool	Int	Real	Str	
Ack	This property can have two values: This works as a global acknowledgement for the tag and is only set to 0 if all alarms for the tag have been acknowledged. 0: There are no alarms associated with this tag that require acknowledgment. 1: There is at least one alarm associated with this tag that requires acknowledgment.	R	Boolean	Y	Y	Y	N	N
UnAck	This property can have two values: If this value is manually set to 1, active alarms (if any) are acknowledged. The value of this property is always the opposite of the Ack property. 0: There is at least one alarm associated with this tag that requires acknowledgment. 1: There are no alarms associated with this tag that require acknowledgment.	R/W	Boolean	Y	Y	Y	N	N
AlrAckValue	Text associated with the Acknowledged state of a Boolean tag. This text is displayed in the Value column of an alarm/event control on page 262. This text can also be edited in the Tag Properties dialog on page 126 (Alarms – Bool Type).	R/W	String, up to 32 chars	Y	N	N	N	Y
AlrOffValue	Text associated with the Normalized state of a Boolean tag. This text is displayed in the Value column of an alarm/event control on page 262. This text can also be edited in the Tag Properties dialog on page 126 (Alarms – Bool Type).	R/W	String, up to 32 chars	Y	N	N	N	Y
AlrOnValue	Text associated with the Active state of a Boolean tag. This text is displayed in the Value column of an alarm/event control on page 262. This text can also be edited in the Tag Properties dialog on page 126 (Alarms – Bool Type).	R/W	String, up to 32 chars	Y	N	N	N	Y
AlrDisable	This property can have two values: 0: The alarms associated with this tag are enabled. This means that if an alarm condition occurs, the alarm becomes active. 1: The alarms associated with this tag are disabled. This means that even if an alarm condition occurs, the alarm does not become active.	R/W	Boolean	Y	Y	Y	N	N

Working with Tags



- If a property is marked "n/a" with regards to being retentive, it is because the property is inherent in the tag definition (e.g., Name, Size) or the value of the property is continuously derived at runtime (e.g., alarm activation, DisplayValue). To enable retention for a tag, select the **Retentive Parameters** option in the [Tag Properties dialog on page 124](#).
- If the project attempts to write a value outside of the array specified in the **Min** and **Max** properties, the tags database will not accept the new value and a warning message is written in the [Output on page 85](#) window. If both **Min** and **Max** properties are configured with the value "0" (zero), it means that any value applied to the tag type is written to the tag.
- Tag properties (such as bit fields) cannot be used to configure [alarm on page 247](#) or [trend on page 269](#) worksheets.
- Although tag properties can be applied to [system tags on page 110](#), those properties are not maintained if the project is downloaded to a CE device.

7.6 Using Tags in the Project

Once a tag has been added to the project database it can be used in the project by associating it to objects on a screen.

The basic process for associating tags to screen objects includes the following steps:

1. In the WinStudio screen, select the object to which the tag is to be applied.
2. Click one of the buttons in the [Animations group on page 220](#) to apply that animation to the object.
3. Double-click on the object to open its [Object Properties dialog on page 164](#).
4. Locate the **Tag** text box for that property and enter the tag name into the field. Tag text box names and locations vary, depending on the type of property that is used.

For example:

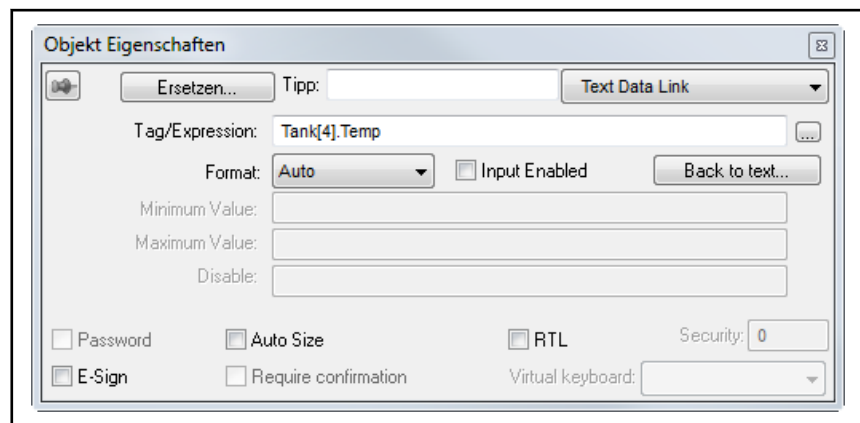


Fig. 7-16: Applying tags to an object

Comprehensive instructions for applying tags to screen objects are provided throughout the documentation where applicable.

7.7 Deleting a Tag from the Project Database

No longer required tags can be deleted by deleting its line in the "Application Variables" datasheet.



Before deleting a tag, it is highly recommended to use the [Cross Reference List on page 87](#) tool to make sure the tag is not being used anywhere in the project.

(If a tag is deleted that is still being used, the project cannot be verified and executed.)

Correct any screens or worksheets where the tag is being used before proceeding.

1. [Stop the project on page 60](#) if it is running.
2. Open the [Project Tags datasheet on page 106](#).
3. Right-click anywhere in the datasheet and then select **Disable Sort** from the shortcut menu. Tags cannot be deleted if the datasheet is sorted in any way. The tags are returned to their default order, i.e., the order in which they were created.
4. Search the line for the tag to be deleted in the datasheet
5. Right-click the line and then select **Delete Line** from the context menu. (If the option is not available, sorting needs to be disabled as described above.) A warning message is displayed to confirm this action
6. Click **Yes**. The line is deleted from the datasheet.
7. Save and close the datasheet.

7.8 Renaming and Removing Tags

7.8.1 Replace Tags Globally

When clicking on the **Global Replace** tool from the Tag Properties toolbar in WinStudio, the following window is displayed:

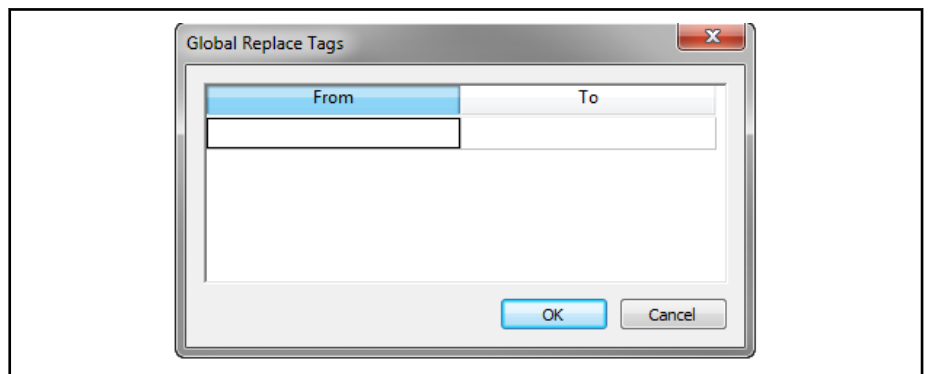


Fig.7-17: Global Replace dialog

From the "Global Replace" dialog, any tag(s) from any document (screens and worksheets) of the whole project can be replaced. Both the **From** and the **To** column can be replaced.

When replacing compound tags (array size > 0 and/or type = class) a specific array position (for example, **TagA[1]**) or class element (for example, **TagB.MemberX**) or both (for example, **TagC[3].MemberY**) can be configured.

Working with Tags

If only the main tag name (for example, **TagC**) in the **From** column is configured, all tags from this main tag are modified for the tag configured in the **To** column.

If an invalid replacement value is configured (for example, replace the **Main Tag** tag from a class-type tag for a simple tag (not a class tag)), the **OK** button is disabled.

When the **OK** button is pressed, the tags configured on the "Global Replace" dialog are replaced in the order that they were configured on the dialog interface.



All documents have to be closed (screens and worksheets) before executing this command.

When changing the tag name on the [project tags database worksheet on page 106](#), "WinStudio" asks whether this tag is to be replaced through the whole project.

The **Replace** option is provided in the **Edit** menu. Use this option to open the "Global Replace" dialog. However, changes are only applied to the screen or worksheet currently active.

7.8.2 Replacing Tags in a Document or Screen Object

To replace all tags in a screen object, double-click the object to open its "Object Properties" dialog and then click **Replace**.

To replace all tags in the current document, carry out one of the two following actions:

- Open the "WinStudio Screen" menu in the IndraWorks main menu and select the entry **Replace tags and strings in the current document**
- Display the "WinStudio Table Functions" toolbar and click on the **Replace tags and strings in the current document** button.

In both cases, the "Replace" dialog is displayed as described below.

One or more tags can be replaced by clicking the **Whole Tag Name** tab. The left column, **From**, displays the tags currently used. Into the right column, **To**, the new names can be entered.

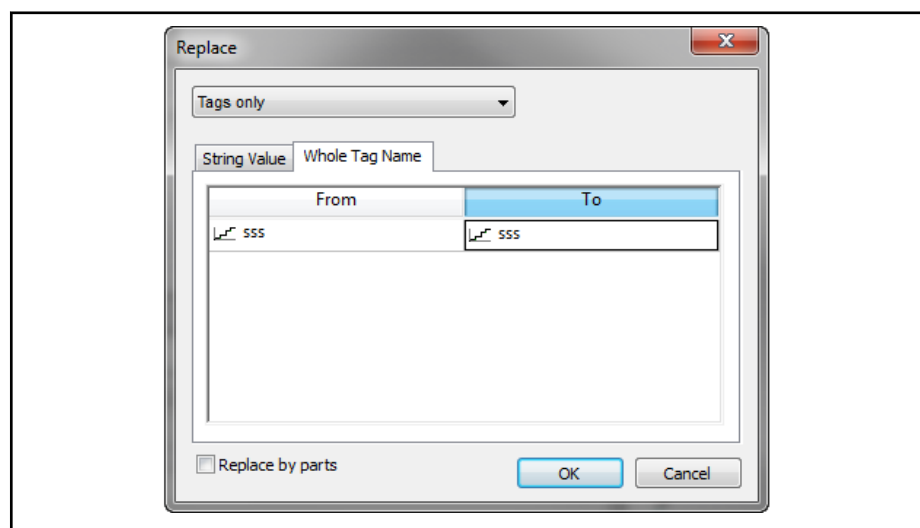


Fig. 7-18: "Replace" dialog, "Whole Tag Name" tab

Note that this does not "rename" or "delete" any tag. Tags used in the object are only replaced by other tags from the database.

Working with Tags

By activating the "Replace by parts" option, replacing parameters (e.g. array index) within classes and arrays becomes easier.

One or more strings can also be replaced (e.g., button captions, descriptive texts) by clicking the **String Value** tab.

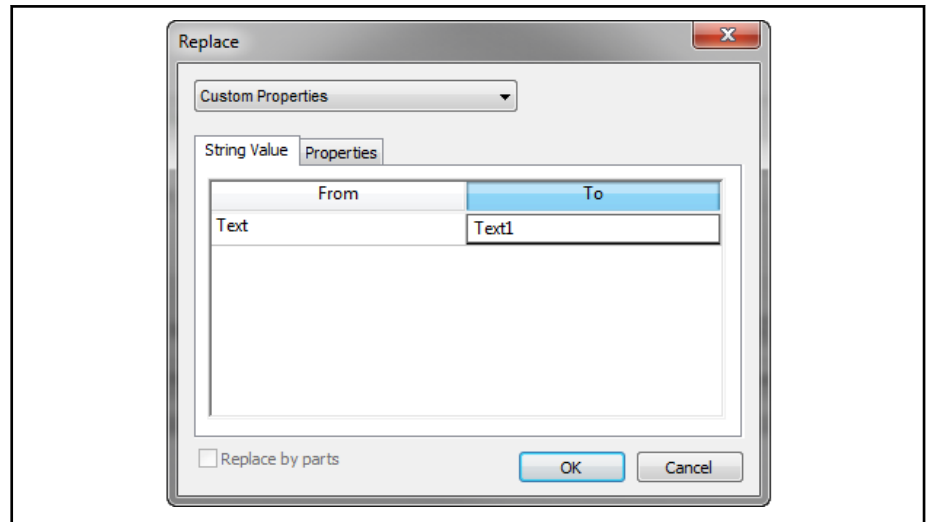


Fig.7-19: "Replace" dialog, "String Value" tab

After editing all entries, click **OK**.

7.8.3 Removing Unused Tags from the Project Database

The **Remove unused tags** tool is used to scan the project database for unused tags to select and remove them.

"Unused tags" are tags defined in the project database that have not been used in any screen or task worksheet.

Since the project has a limited number of available tags (as determined by the product/license type), some or all of these unused tags can be removed to decrease the project tag count.

To search for unused tags and remove them, proceed as follows:

1. Save and close all open project screens and worksheets.
2. Execute the "Verify Application" function in the context menu of the "WinStudio" folder and press the "Verify" button in the "Verify Application" dialog.

The project is automatically verified by the development application. If unused tags are found, a message is displayed in the Messages window.

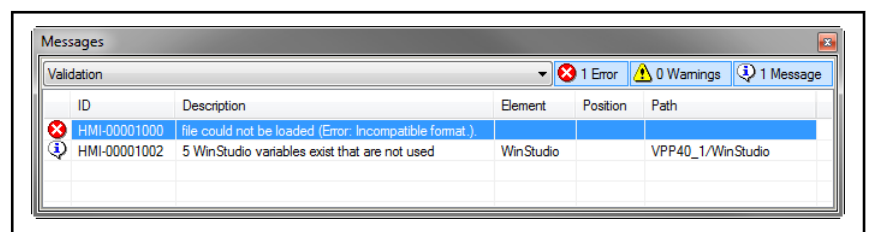


Fig.7-20: Information on unused tags on "Verify Application"

Double-click on the respective line in the Messages window. The following dialog is displayed:

Working with Tags

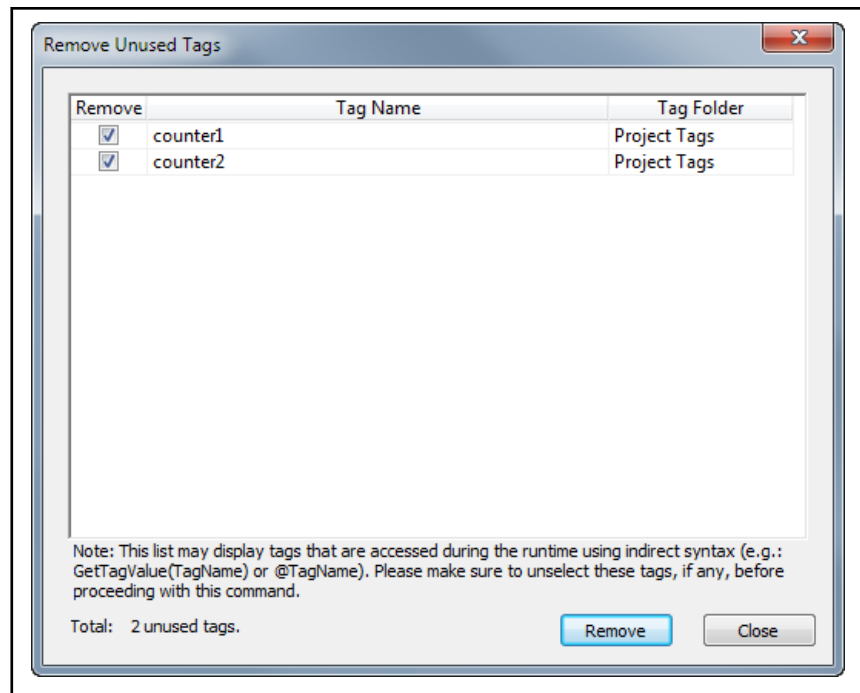


Fig.7-21: List of unused tags in the "Remove Unused Tags" dialog

3. Determine which tags are to be removed, if any

- If all of the listed tags are to be removed, click **Remove**.
- To keep some of the listed tags, clear the "Remove" option for these tags and then click **Remove**.



The listed tags may include some that are accessed at runtime using indirect syntax (e.g., **GetTagValue(TagName)** or **@TagName**, where the value of **TagName** is the name of an unused tag).

- If none of the listed tags are to be removed, click **Close**.

WinStudio removes the selected tags and then asks to verify the project again

4. Click **Yes** to verify the project again

7.8.4 Tags Database Reset

Execute the function **Reset Tags Database** to reload the tags database on the local station.

This command affects all tags stored in the "Application Variables" folder.

This option is useful for resetting the application variables and restoring the values they had when the project was loaded for the first time.

If the project is stopped while the development environment remains open, the tags are not reset by default if the project is executed again. Therefore, this command can be used to reset these tags before the project is executed again.

If this command is executed, the **Startup Value** configured for each tag ([Tags Properties dialog on page 140](#)) is entered into the respective tag.

If no **Startup Value** has been configured for a numeric tag (**Boolean**, **Integer** or **Real**), the value "0" (zero) is entered into the tag.

Working with Tags

If no **Startup Value** has been configured for a String tag, the empty value (" ") is entered into the tag.

This command is disabled (in gray) if there is at least one runtime task running on the local station.

Before executing this command, all runtime tasks have to be closed (**Stop application** in the context menu of the "WinStudio" folder).



The tags in the "Internal Variables" and "Shared Variables" folders are not affected by this command.



If the application variables are to be reset automatically on running the project (**Start application** in the context menu of the "WinStudio" folder), the option **Reset tags database when starting project** on the [Preferences tab of the Project Settings dialog on page 76](#) can be selected.

7.8.5 Using Available Tags

Click the **Object Finder**  to open the "Object Finder" dialog, which lists all **Tags** and **Functions** currently configured for the project.

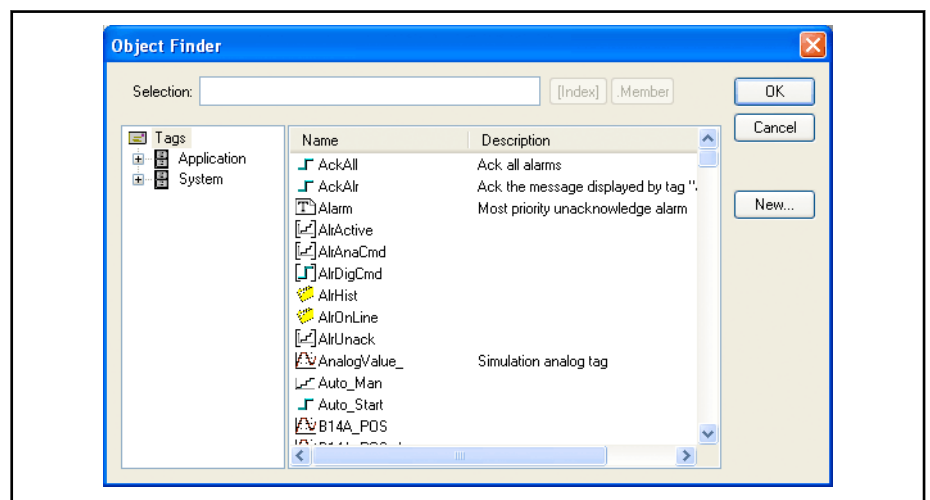


Fig. 7-22: Object Finder dialog

To select an existing tag/function, double-click on the tag/function name, and then click **OK** to close the box. The selected name is displayed in the **Tag name** text box.

- To select a specific array index, click the **Index** button after specifying the [Array Tag on page 114](#) name.
- To select a specific element of a class tag, click the **Element** button after specifying the [class tag on page 108](#) name.
- To create a new tag, click the **New** button.
- When the "New Tag" dialog is displayed, enter the following information and click **OK** to close the box.
 - **Name**
 - **Array size**
 - **Type** (Boolean, Integer, Real, String, Class:Control, Class:msgonline or Class:Alr)
 - **Description**

Working with Tags

- **Scope** (local or server)

7.8.6 Changing Tag Properties

The "Tag Properties" dialog can be accessed via "WinStudio Cross References".

Click the **Tag Properties** tool to configure parameters for each tag.

(For further information on specifying tag properties, refer to [understanding tag properties and parameters on page 124.](#))

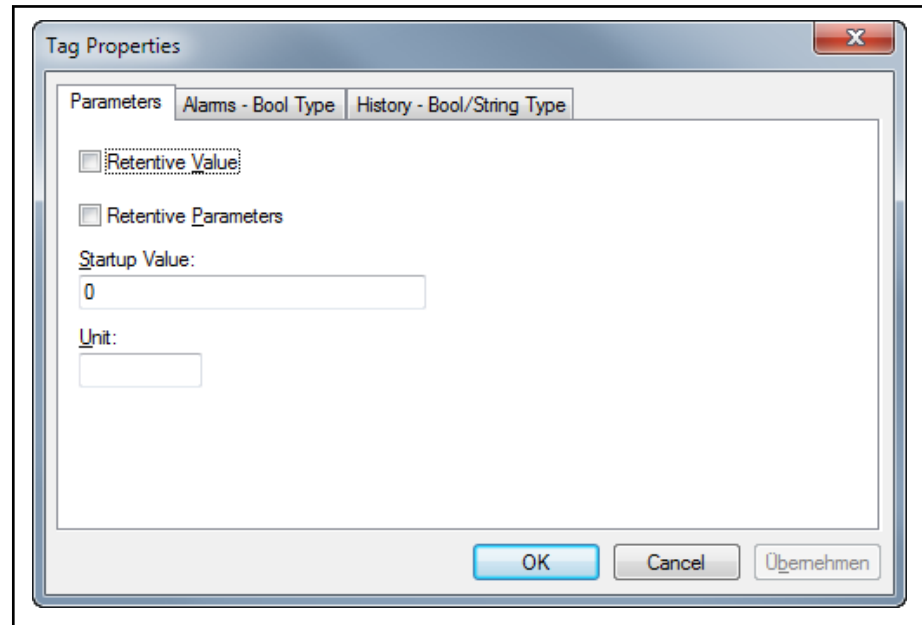


Fig. 7-23: Example for Tag Properties dialog



The "Tag Properties" dialog that displays for Boolean, Integer, Real, and String tags differs in content.

7.9 Importing an External Database

7.9.1 Import Wizard

The Import Wizard is a powerful tool that reduces engineering time during project development.

The Import Wizard enables import of tags from different data sources directly to the project tags database. Depending on the data source, not only tag names can be imported but also the communication interface (the link between the tags and the PLC addresses).

Select the **Variable Import Wizard** in the "Tools" context menu of the "WinStudio" folder to display the "Import Database Wizard" dialog as step-by-step guide through the process of importing tags.

Three steps for importing tags from these data source types are necessary:

- WinStudio Project Database
- OPC Server Database
- CSV Database
- ODBC Database

1 Step: Select the Source Type

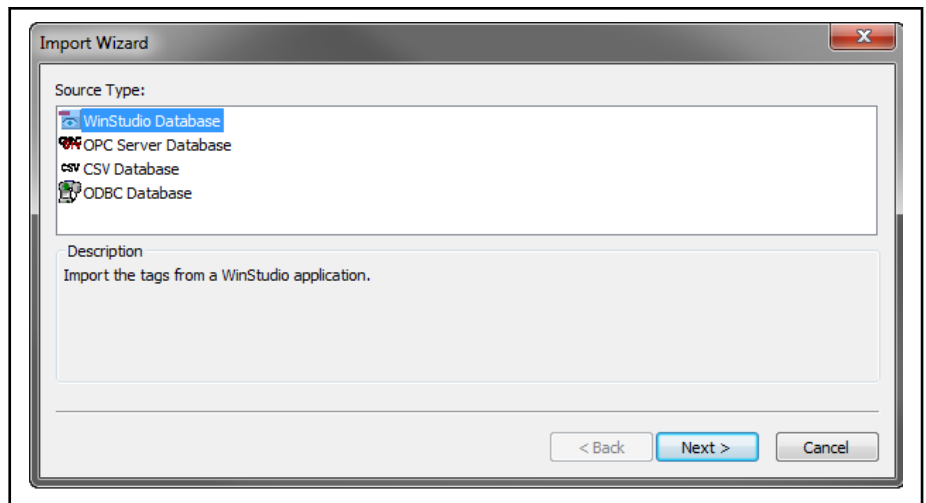


Fig. 7-24: Import Wizard

Click on the data source type from which the tags are to be imported. Click on **Next**.

Continue with the respective section of the instructions required to complete the database import: ["Data source types" on page 143](#)

2 Step: Configure the Source Type Settings

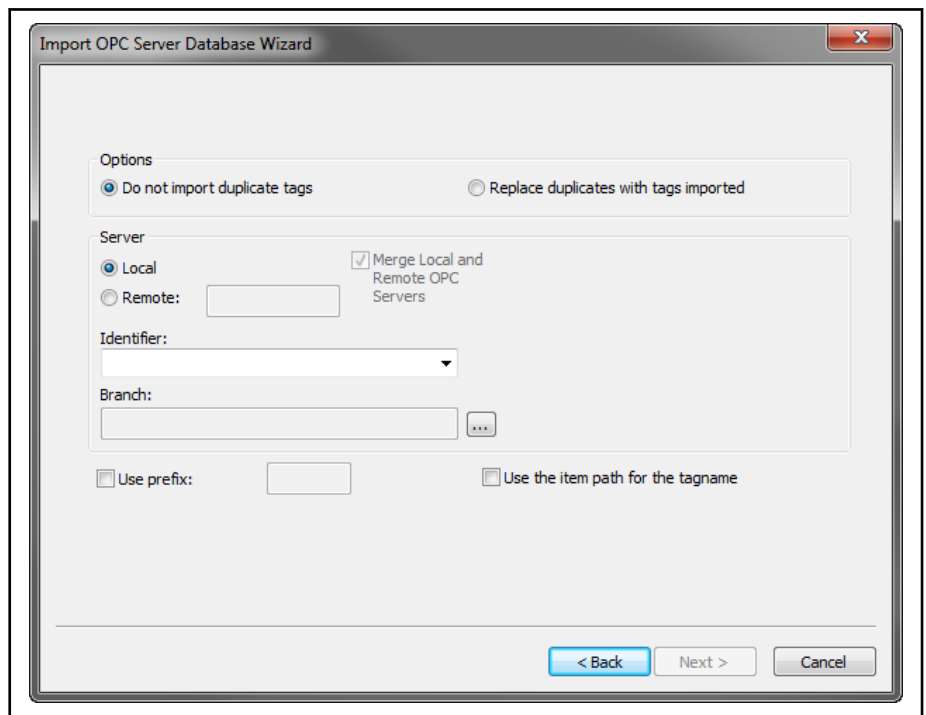


Fig. 7-25: Import OPC Server dialog

Most of the settings in the second window depend on the data source type selected in the first step. The screenshot above is an example of one data source type (OPC Server Database). The settings that are common for any data Source Type are described below:

- **Options:** Select **Do not import duplicate tags** to prevent that imported tags overwrite tags with the same name that already exist in the tags database of the current project.

Select **Replace duplicates with tags imported** to overwrite tags in the Tags Database with imported tags of the same name.

Working with Tags

- **Use prefix:** Enable to specify a prefix (up to 4 characters) to be added in front of the name of the imported tags.

It is useful to use a prefix to differentiate the imported tags from the tags created manually.



The other settings vary according to the data source selected in the first step and they are described in the specific sections for each ["Data source types"](#) on page 143.

After configuring the settings in this dialog, click **Next**.

3 Step: Filter the tags

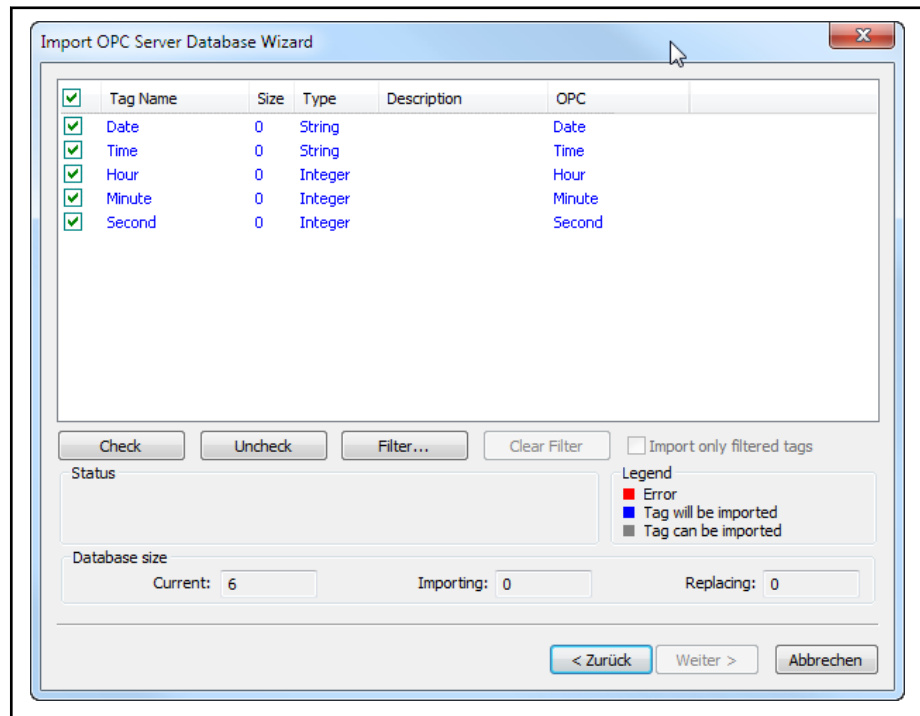


Fig.7-26: Import OPC Server dialog

The preceding figure is an example of one data source type (OPC Server Database). The fields and settings that are common for all data source types include the following:

- **Grid:** Displays the list of tags found on the data source.
 - **Checkbox:** Enable to import the tag from the data source to the tags database of the current project.
 - **Tag Name:** Name of the tag
 - **Size:** Array size of the tag
 - **Type:** Data type of the tag (Boolean, Integer, Real, String or Class:<ClassName>)
 - **Description:** Description of the tag
- **Check:** Click to select/import all tags in the grid
- **Uncheck:** Click to uncheck all tags in the grid
- **Filter:** Click to filter the tags. The Filter dialog is displayed on which a mask can be specified for each column in the grid. Placeholders (* and ?) can be used to filter data.
- **Clear Filter:** Click to reset the filter.

Working with Tags

- **Import only filtered tags:** Check this option to import only the tags that are displayed in the grid (filtered).
- **Status:** Displays a message describing the status of the tag currently selected in the grid. This information is especially useful to indicate why a tag cannot be imported.
- **Legend:** Describes the meaning of the colors that indicate the tag status:
 - (Red) **Error:** Tag cannot be imported because it is not supported by "WinStudio". Refer to the text field for a detailed description of the error.
 - (Blue) **Tag will be imported:** The tag will be imported after clicking the Finish button.
 - (Gray) **Tag can be imported:** The tag can be imported but it has not been selected.
- **Database size pane:** Displays summary information regarding the current Import Wizard:
 - **Current:** Displays the current number of available tags in the "Application Variables" datasheet.
 - **Importing:** Indicates the number of tags selected to be imported
 - **Replacing:** Indicates the number of tags configured in the project tags database of the current project that will be replaced by an imported tag with the same name.

After selecting the tags to import, click the Finish button or click Cancel to abort the operation.

The other settings vary according to the data source selected in the first step and they are described in the following sections for each data source type.

Data source types

Continue with the respective section for the instructions to complete the import database procedure:

- [Importing from Other Project Databases on page 143](#)
- [Importing from OPC Server Databases on page 145](#)
- [Importing from CSV Databases on page 146](#)
- [Importing from ODBC Databases on page 147](#)

7.9.2 Importing from...

Importing from Other Project Databases

This wizard allows to import the objects (tags and worksheets) of other "WinStudio" projects.

If only the tags (rather than the whole project) are imported from a remote computer, the TCP/IP client worksheet can be automatically created to link the tags between both stations (the local and the remote) and to share the value of these tags between both stations at runtime.

Working with Tags

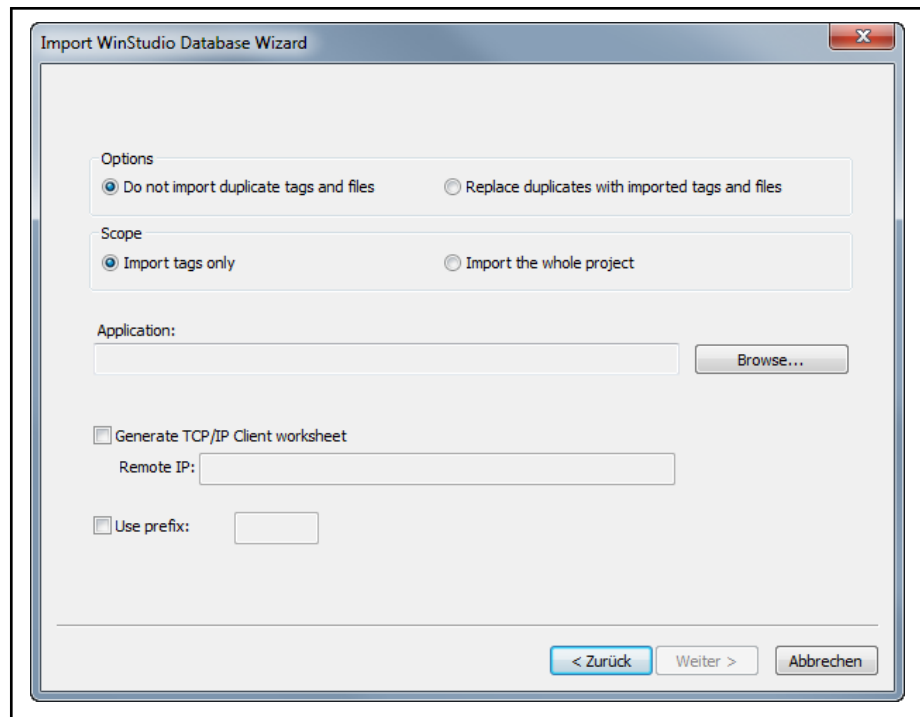


Fig. 7-27: "Import WinStudio Database Wizard" dialog

- **Import tags only:** If this option is selected, the tags of the other project are imported to the current project. The other elements of the project (worksheets) are not imported.
- **Import the whole project:** If this option is selected, the following elements of the other project are imported to the current project.
 - Tags database
 - Global procedures
 - Screens
 - Screen groups
 - Web pages
 - Alarms
 - Trends
 - Recipes
 - Reports
 - ODBC
 - Math
 - Scripts
 - Scheduler
 - Drivers
 - OPC
 - TCP/IP
 - DDE

This option is useful for merging projects and importing template projects.



When selecting the option to **Import the whole project**, the following worksheets are always imported, regardless of existing worksheets with the same number in the current project: ODBC, Math, Script, Scheduler, Drivers, OPC, TCP/IP and DDE.

If there are worksheets with the same number in the current project, worksheets imported from the other project are inserted as additional worksheets in the current project and the number of each worksheet is automatically increased to avoid replacing files on the current project.

- **Do not import duplicate tags and files:** If this option is selected, the following elements are not imported in case there is already an equivalent element in the current project:
 - Tags database (tags with the same name are not imported)
 - Global procedures (the global procedures are not imported)
 - Screens (screens with the same name are not imported)
 - Screen groups (screen groups with the same name are not imported)
 - Web pages (Web pages with the same name are not imported)
 - Alarms (alarms assigned to tags with the same name are not imported)
 - Trend (trend logs assigned to tags with the same name are not imported)
 - Recipes (recipes with the same name are not imported)
 - Reports (reports with the same name are not imported)
 - Script (the startup script is not imported)
- **Application:** Click **Browse...** and select the **project_name.APP** file with the tags to be imported.
- **Generate TCP/IP client worksheet:** If the project is to share tags with another "WinStudio" project running on a remote server, select this option and enter the IP address of the respective server. "WinStudio" automatically configures the TCP/IP client worksheet to exchange data with the remote project.
- **Use prefix:** Select this option to specify a prefix (up to 4 characters) that is added in front of the names of the imported tags. It is useful to differentiate the imported tags from the tags created manually.

Importing from OPC Server Databases

This wizard enables tag import from either a local OPC server or a remote one.

If tags are imported from the OPC server, the OPC client worksheet is created automatically to link the tags. For this reason, the communication interface between the "WinStudio" OPC client and the external OPC server does not need to be configured.

Working with Tags

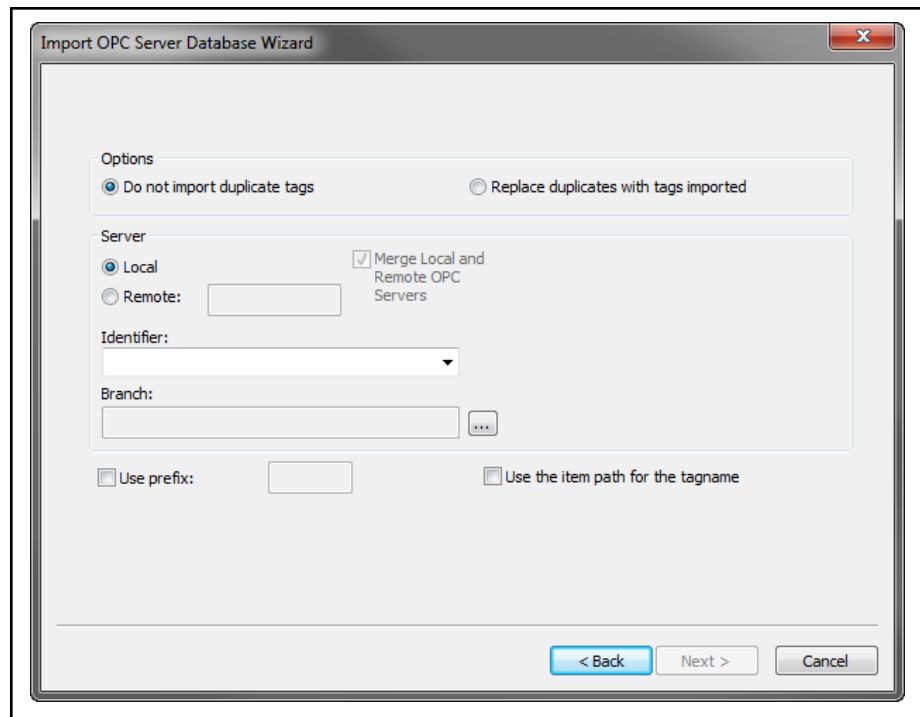


Fig.7-28: Import OPC Server Database Wizard

- **Local/Remote:** Provide the following options:
 - **Local:** Select this option to import tags from an OPC server installed on a local computer.
 - **Remote:** Select this option to import tags from an OPC server installed on a remote computer. Enter the IP Address (or the host name) of the remote computer on which "WinStudio" is running in the **Remote** field.
- **Merge Local and Remote OPC Servers:** If a remote server is selected, select this option to display the list of OPC servers installed on the local and remote computer. Uncheck this checkbox to display only the list of OPC servers installed on the remote computer.
- **Identifier:** Displays the list of available OPC servers.
- **Branch:** Click on the Browse button (...) to select the branch of the OPC server from which the tags (items) are to be imported. Leave this field blank to import tags from all branches configured on the OPC server.
- **Use the item path for the tagname:** Check this option to add the path name to the item name when importing tags from the OPC server. Uncheck this option to use only the item names configured on the OPC server.

For this data source type, the table displayed in step 3 of the **Import Wizard** (in the context menu of the "WinStudio" folder) offers an additional **OPC** field displaying the item name on the OPC server.



Refer to Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common for all source types.

Importing from CSV Databases

This wizard enables importing tags from a text file into the CSV (Comma Separated Values) format or any similar format.

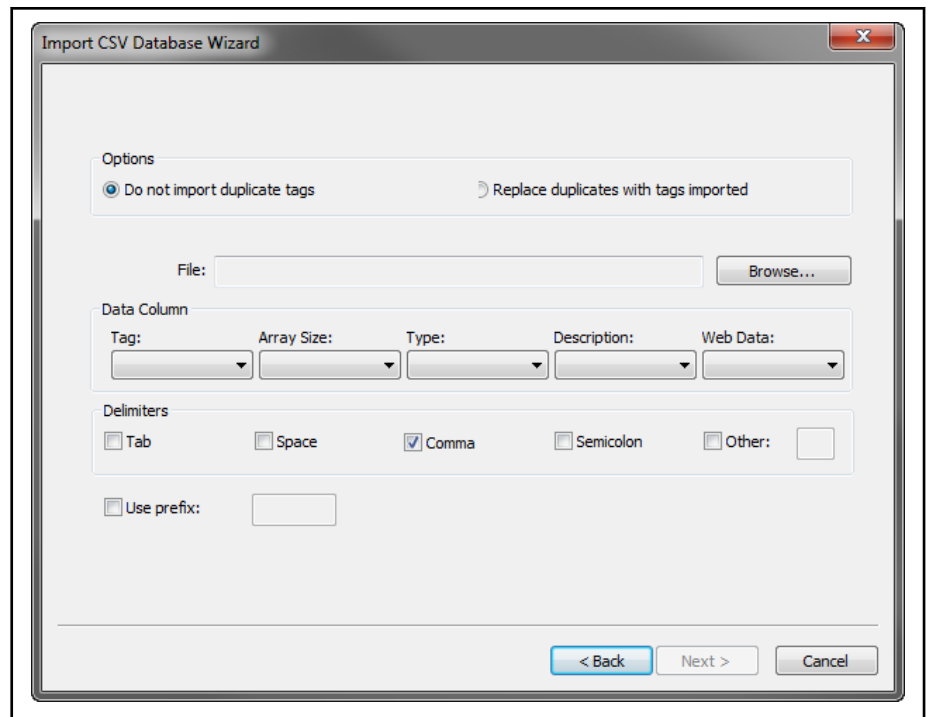


Fig. 7-29: Import CSV Database Wizard

- **File name:** Click the "Browse" button to select the text file from which the tags are imported.
- **Data Column:** Select a number for each tag property that corresponds to its column number in the import file.

For example, if the Tag, Array Size and Type are listed in the second, third and first columns in the import file, respectively, select 2 in Tag, 3 in Array Size and 1 in Type. The Tag property (tag name) is mandatory but the other properties are optional.

For properties that are not included in the text file, select the option **Not used**. "WinStudio" inserts defaults or leaves the field blank, according to the following list:

- **Array size:** 0
- **Type:** Integer
- **Description:** <Blank>
- **Web Data:** Local

- **Delimiters:** Select the delimiter(s) used in the text file to divide one column from another. For a CSV file, the delimiter is Comma (the default). More than one delimiter can be selected simultaneously and a custom delimiter can be entered by using the option "Other".



Refer to Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common for all source types.

Importing from ODBC Databases

This wizard enables importing tags from an external SQL relational database such as Microsoft Access, SQL server, Oracle, My SQL, Sybase and others via the ODBC interface.

Working with Tags

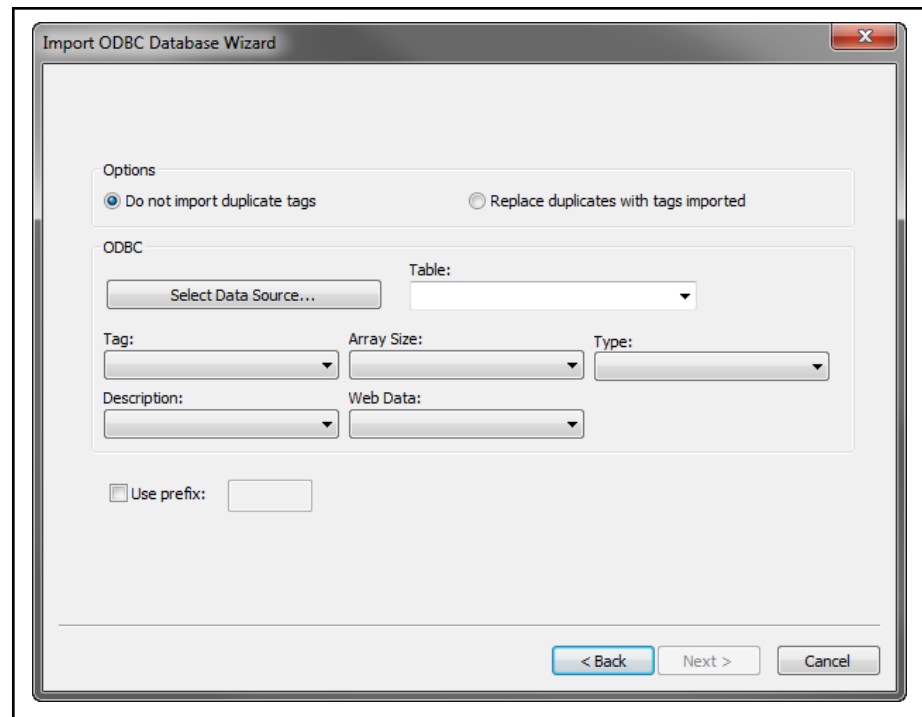


Fig.7-30: Import ODBC Database Wizard

- **Select Data Source:** Click to select the ODBC Data Source Name (DSN) linked to the database from which the tags are to be imported. The DSN has to be created before by means of the Data Sources (ODBC) window (**Control Panel ► Administrative Tools ► Data Sources [ODBC]**).
After selecting a DSN, the other fields in this window are automatically filled with information from the selected database.
- **Table:** Select the table that holds the tags in the import database.
- **Tag:** Select the name of the column that holds the tags in the import database.
- **Array Size:** Select the name of the column that holds the array size for the tags in the import database.
- **Type:** Select the name of the column that holds the tag type in the import database.
- **Description:** Select the name of the column that holds the tag description in the import database.
- **Web Data:** Select the name of the column that holds the Web Data for the tags in the import database.



Refer to Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common for all source types.

7.10 Tag Integration

7.10.1 Tag Integration, General Information

With the automatic tag integration function, variables configured in control systems can be directly selected in the WinStudio variable browser (Object Finder) without having to create a respective tag and configuring a driver sheet first.

Working with Tags

This is different from the other options to include third-party tags in the project, such as importing the tag definitions or configuring a worksheet to associate project tags with external items.

With tag integration, the third-party tags are "live", i.e., they are continuously and bilaterally updated at runtime as long as the third-party control application or device are also running, and they can be used anywhere in the project where project tags would normally be used.

There are two types of tag integration:

1. **Integrated:** All tags from the source can be browsed in the Object Finder, but individual tags are neither added to the project tags database nor counted against the project tag limit until they are actually used somewhere in the project.
2. **Shared:** All tags from the source are automatically added to the project tags database and counted against the project tag limit.



This option is obsolete and available for compatibility with older projects. If this type is to be used in a new project, please contact Customer Support for support.

Tag integration is part of the project settings.

- [Integrating tags from IndraLogic on page 149](#)

WinStudio can connect to and share tags on Bosch control systems and IndraLogic devices.

- Using the Object Finder to select integrated tags

After adding a tag integration source to the project, the tags from that source can be viewed and selected in the Object Finder like any other project tag.

- Renaming integrated tags in the project

If WinStudio integrates tags from control systems, the variable names cannot be applied without changes. Some changes are made to improve tag management and to adhere to the local tag name syntax.

7.10.2 Sharing Tags on IndraLogic Devices

WinStudio can connect to and share tags on Bosch control systems and IndraLogic devices.

Automatically sharing tags via IndraWorks

If WinStudio is executed along with Rexroth IndraWorks on the same station, all device communication and tag sharing is managed by IndraWorks. No configuration in the development application is required. WinStudio automatically connects to all registered devices and makes their tags available for use in the project. The tags are dynamically and continuously updated and can be browsed using the [Shared Database on page 109](#) folder or the [Object Finder on page 139](#).

Working with Tags

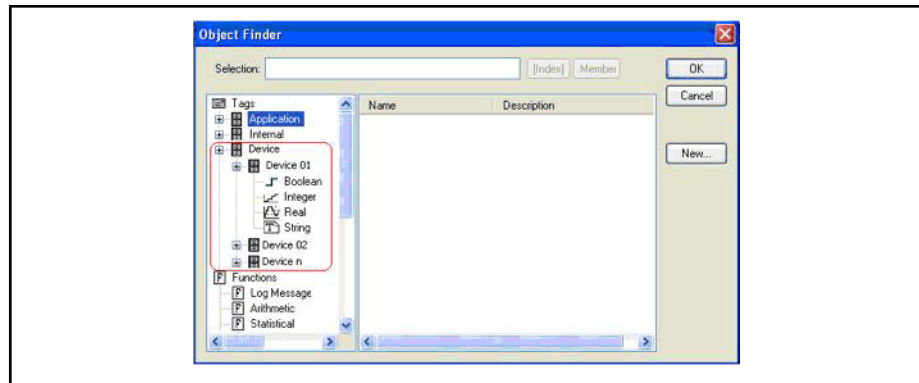


Fig. 7-31: Browsing devices in the Object Finder

The tags are grouped by device in the "Object Finder" to make them easier to browse, and they are also categorized by data type to ensure compatibility with WinStudio objects and functions. These shared tags can be used in the project just like any other project tag.

For further information about using Rexroth WinStudio and configuring IndraLogic devices, please refer to the "IndraWorks Operation and Programming Guide".

Configuring WinStudio to communicate with devices

If the WinStudio project and the used control system are managed together in one IndraWorks project, no further settings are required.

If variables from control systems are to be used that are not included in the project, proceed as follows to add them:

1. In the Project Settings dialog of the WinStudio folder, open the **Communication** tab.

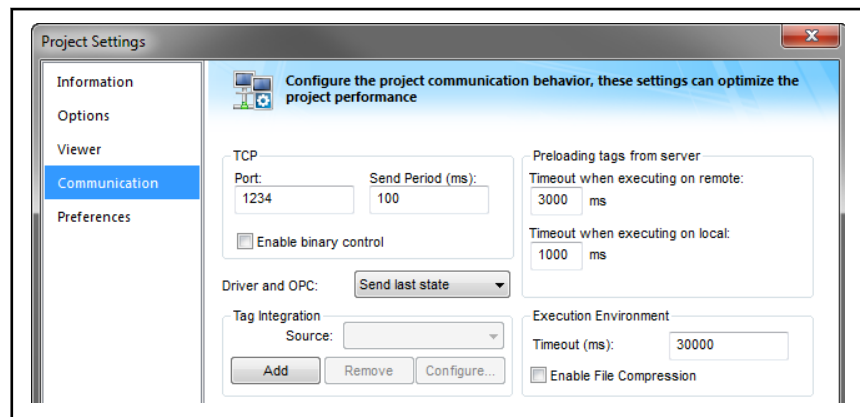


Fig. 7-32: Communication tab of the Project Settings dialog

2. In the "Tag Integration" pane, click **Add**. The "Tag Integration Source" dialog is displayed:

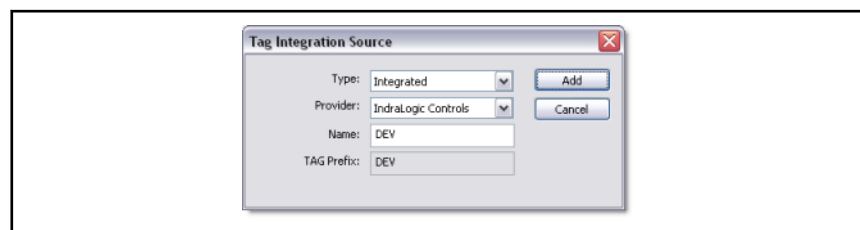


Fig. 7-33: Selecting the source for tag integration

3. In the **Type** list, select **Integrated** if it is not already selected.

Working with Tags

4. Select the communication type supported by the control system from the **Provider** list.
5. In the **Name** field, type a name for the target device. This is the name that is displayed in the "Object Finder" and under which the tags are grouped. The name has to follow the proper tag name syntax. Otherwise, WinStudio does not accept it. For further information, refer to "[Understanding the tag name syntax](#)" on page 111. WinStudio automatically duplicates the device name entered into the **TAG Prefix** field. This prefix is added to the local project tags that are created to synchronize with the device tags. (Refer to "How device tags can be renamed in WinStudio" below.)
6. Click on **Add**. A configuration dialog is displayed to adjust special settings for the selected provider.

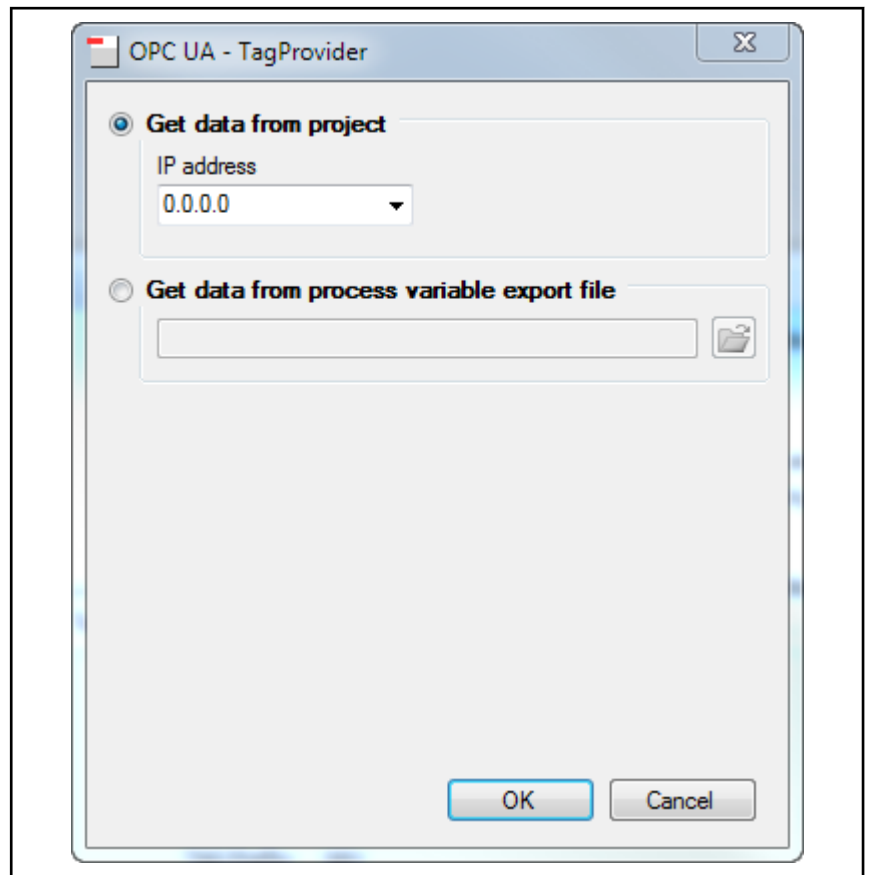


Fig.7-34: Variable provider configuration for OPC UA communication

7. Click on **OK**.

How device tags can be renamed in WinStudio

If WinStudio creates local project tags to correspond to the device tags, the device tag names cannot be applied directly without changes. Some changes are made to improve tag management and to adhere to WinStudio's tag name syntax.

The full, original name of a device tag can always be retrieved by accessing the "DeviceTag" tag field on the corresponding project tag.

For example, **tagName->DeviceTag**.

Inserting the tag prefix

Working with Tags

First of all, since the project may connect to multiple devices that have the same control program and device tags, the development application automatically inserts the device name / tag prefix specified when adding the device.

For example, for a device tag named...

MyDeviceTAG[1].ClassMember

...the corresponding WinStudio project tag is named...

device_name.MyDeviceTAG[1].ClassMember

This allows to differentiate between similar tags on different devices.

Multidimensional arrays

WinStudio does not support multidimensional arrays. For this reason, if device tags have more than one array index, each index after the first will be indicated with **_Index_**.

For example, for a device tag named...

MyDeviceTAG[1][2][3].ClassMember

...the corresponding WinStudio project tag is named...

device_name.MyDeviceTAG[1]_2_3_.ClassMember

Nested classes

WinStudio does not support nested classes. For this reason, if device tags have more than one class member, each class member after the first will be represented with **_ClassMember**.

For example, for a device tag named...

MyDeviceTAG.ClassMember.ClassMember2

...the corresponding WinStudio project tag is named...

device_name.MyDeviceTAG.ClassMember_ClassMember2

Refreshing tag information from devices

In the "Object Finder", configured devices can be displayed with three different symbols:

Symbol	Description
 DEV	Device tag information was extracted from a symbol file and then cached.
 DEV	Device tag information is currently being received from a connected device.
 DEV	Connection to device has failed. Tag information might not be up-to-date.

To refresh the device tag information, either by rereading the symbol file or by trying to reconnect to the device, click **Refresh Devices** in the "Object Finder". All configured devices are refreshed at the same time.

Changes in the device tag information might break tag references in the project. To find broken tag references, verify the project (**Verify** on the Home tab of the ribbon).

Synchronizing the project tags database with the device

If a target device is added to the project, the development application reads the device symbol file and adds corresponding tags to the project tags database. After that, the database has to be synchronized to reflect any changes in the device tags. Proceed as follows to synchronize the database:

1. If the **Automatic Synchronization** option is selected for the target device, the database is automatically synchronized whenever the project is veri-

Working with Tags

fied (**Verify** on the Home tab of the ribbon) or the tags are browsed using the Object Finder.

In this case, WinStudio gets a list of the current tags on the target device and compares it to the tags database. The "Database Synchronization" dialog is displayed, listing any tags that have been deleted from the device since the last sync.

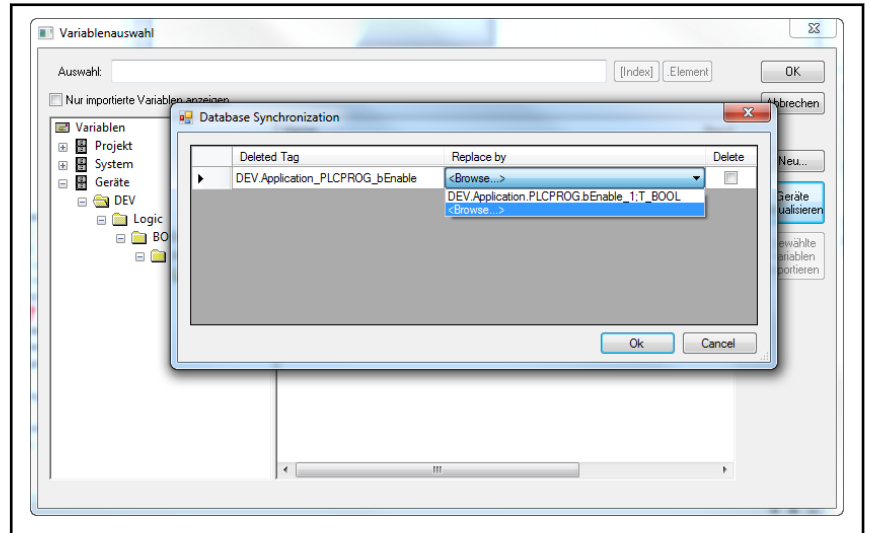


Fig. 7-35: Database Synchronization dialog

For each tag in the list, either a tag from the project can be deleted or all respective tags (in objects and worksheets) can be replaced by a new tag:

- Proceed as follows to replace a tag:
 1. Click on the **Replace by** column for this tag.

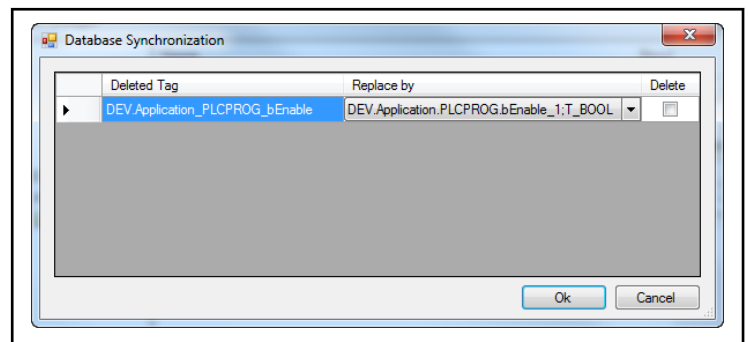


Fig. 7-36: WinStudio, added tags

2. Select an existing variable from the list or click **Browse** to insert a variable.
 3. Click **OK** to close the "Database Synchronization" dialog.
- To delete a tag, click (check) the **Delete** checkbox for that tag. If a tag is deleted from the project while still being used by objects and worksheets, errors will occur the next time the project is verified or executed. The [Cross Reference List on page 87](#) tool can be used to identify the use of tags in the project.
2. Click **OK** to apply the changes and close the "Database Synchronization" dialog.

Please note: If a listed tag is not replaced or deleted, the tag is listed again the next time the database is synchronized.

8 Screens and Graphics

8.1 Screens and Graphics, General Information

The ability to display the status of the process by interacting with instrumentation (or computers), is described as the Human-Machine Interface (HMI).

With "WinStudio", high-resolution process screens can be created for process monitoring.

The WinStudio graphic tools consist of two modules:

- The "screen editor" in the "IndraWorks Engineering" development environment (used to create or import graphics); and
- the "Viewer" to display WinStudio screens in the runtime environment

"Animations" can be used to create dynamic graphic objects or symbols. Animations cause objects and symbols to change appearance to reflect changes in the value of a tag or an expression. WinStudio screens usually include several static and animated objects.

Graphics can be selected as background in WinStudio screens.

For example, the static elements in the following WinStudio screen may be part of a background graphic and animated objects can be used to indicate changes to the illustrated process. This creates the impression of a 3D screen.

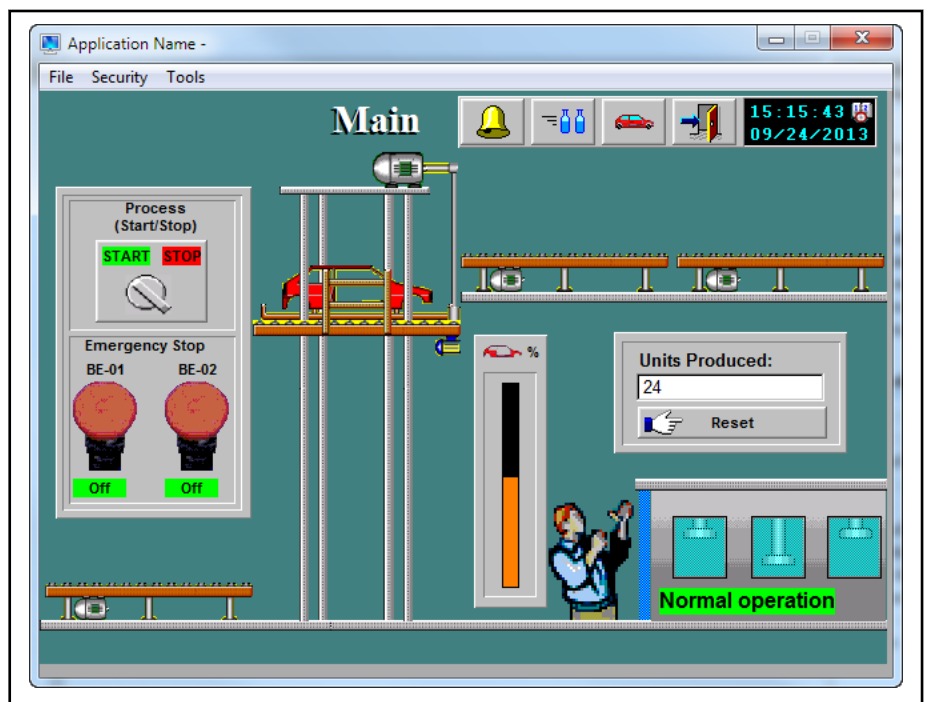


Fig. 8-1: Example: Display of a WinStudio screen in CEView

For configuration of WinStudio applications, a Windows-compatible pointing device, such as a mouse or touch pad, is required.

Project runtime can also be executed in the Viewer without a pointing device if a keypad or keyboard is configured for all commands.

Screens and Graphics

8.2 WinStudio Folders

8.2.1 WinStudio Folder

WinStudio Folder, General Information

Depending on the application type selected on adding the IndraControlIV device, the **WinStudio** folder can be found directly under the "VPx-/VEx/VRx" folder or in the "ScreenApplication" subfolder.

The folder **WinStudio** includes all screens, screen groups and symbols in the project.

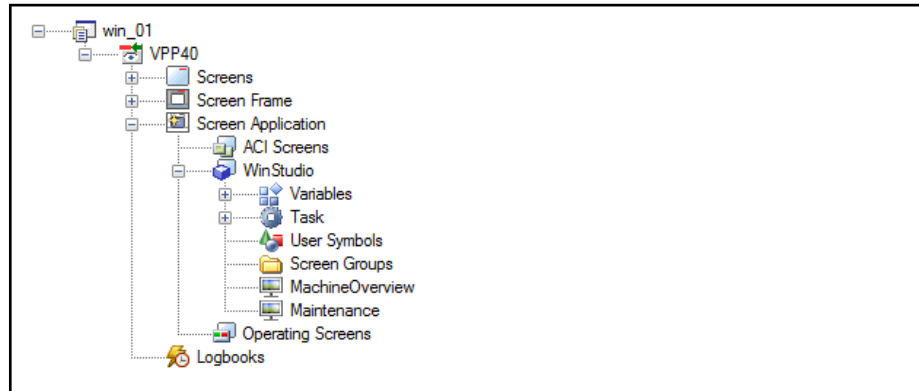


Fig.8-2: The "WinStudio" folder

The individual folders are described on the following pages:

- **WinStudio** contains all [screens on page 156](#) created for the current project
These screens can be structured in a simple list or in user-defined subfolders
- **Variables** contains all functions and editors for editing variable types as well as configuration of process integration of variables into external devices
- **Tasks** contains different editors and functions required to create more complex applications ("math sheets", "alarms", etc.)
- **Screen Groups** contains all [screen groups on page 161](#) (individual WinStudio screens combined into manageable groups) created for the current project
- **User Symbols** contains all [user-defined symbols on page 200](#), which can be groups of screens and/or text. Customized symbols for the project can be created and saved to this folder
- **Graphics Script** contains [predefined functions that are executed when certain screen actions occur on page 718](#), such as when the Thin Client is started on a remote station

Screen Attributes Dialog

The "Screen Attributes" dialog is used to configure runtime settings - such as size, location, title bar, security level, and screen logic - for a specific project screen.

Opening the dialog

To open the Screen Attributes dialog, right-click on the opened WinStudio screen and select "Attributes" in the context menu.

Screens and Graphics

Dialog in detail

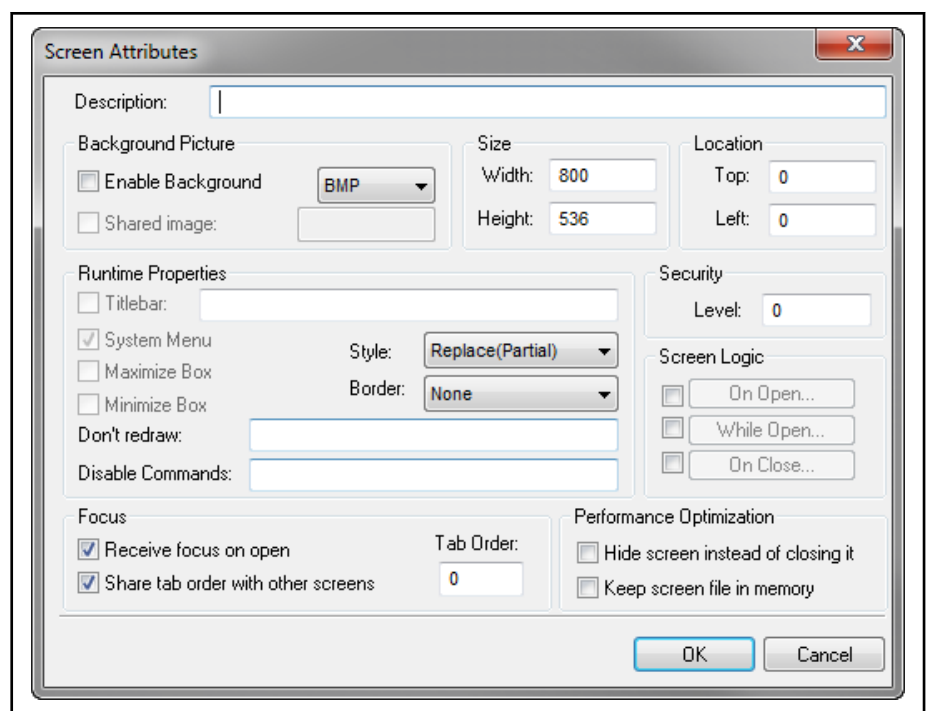


Fig. 8-3: Screen Attributes dialog

"Screen Attributes" dialog elements

Pane / Element name		Description
Description		A short description of the WinStudio screen. This is not shown anywhere at runtime
Background picture	Enable background	Enables the background picture plane and specifies the file type of the background graphic. If this option is selected, a new BMP file is created with the same name as the WinStudio screen (e.g. ..\Screens\<Screen-Name>.BMP). If required, this screen can be edited with a third-party screen editor. For further information, refer to changing the background color or picture of a screen on page 160 .
	Shared image	Uses the stored image file located in the "Screens" subfolder of the project folder. If a tag/expression is to be used instead of a fixed name to change the value during runtime, the tag/expression has to be enclosed in curly brackets. For example: {myTag} Do "not" include the extension in the file name. If the file format is not BMP, use the list to the right of the Enable background option above to select the correct format Note: In projects for the target system Windows CE, only BMP files are supported
Size	Width	The default width of the screen (in pixels) when it is initially displayed. It is possible to change the size during runtime if the screen is set to be resizable. Refer to Table entry below
	Height	The default height of the screen (in pixels) when it is initially displayed. It is possible to change the size during runtime if the screen is set to be resizable. Refer to Table entry below

Screens and Graphics

Pane / Element name		Description
Location	Top	The default distance (in pixels) between the top of the computer display and the top of the screen when the screen is initially displayed. The location can be changed during runtime if "..." is set for the screen. Refer to Table entry below
	Left	The default distance (in pixels) between the left edge of the computer display and the left edge of the screen when the screen is initially displayed. The location can be changed during runtime if "..." is set for the screen. Refer to Table entry below
Security		The minimum security level that a user has to have to access this screen
Performance optimization		Keeps the screen loaded in memory after it is opened the first time, so that it opens more quickly every time thereafter. Any user action or system process that normally close the screen only hides it Caution: This option is only to be selected for critical screens that have to be opened quickly. If too many screens are retained in memory, the overall project performance is affected

Screens and Graphics

Pane / Element name		Description
Runtime Properties	Style	Influences the general runtime behavior of the screen: Overlap: Opens the screen without closing any other screens. Popup: Opens the screen before all other screens as dialog, however, does not close them Replace (partial): Opens the screen and closes all other Replace and Popup windows. This is set by default Dialog: Similar to popup except that the other screens are also disabled until the dialog is closed Replace (complete): Similar to Replace (partial) except that it also closes overlapped windows and dialogs
	Border	The screen border type: None: No border; the screen is a flat, immovable rectangle on the computer display. This is set by default Thin: A thin border that makes the screen a movable window. Includes a title bar Resizing: A thick border that makes the screen a movable, resizable window. Includes a title bar
	Titlebar	Shows the window's title bar with the specified window name If a tag/expression that provides this value is to be specified to programmatically change the value during runtime, the tag/expression has to be enclosed in curly brackets. For example: "{myTag}" Tip: It is useful to specify a window name even when the title bar is not shown, because when the screen is printed, the window name is included in the page header
	System menu	Provides a menu of basic window commands at the left end of the title bar
	Maximize box	Shows the Maximize button at the right edge of the title bar
	Minimize box	Shows the Minimize button at the right edge of the title bar
	Don't redraw	If this tag/expression is set to TRUE, the screen is not updated
	Disable commands	If this tag/expression is set to TRUE, the screen is locked against user action but the graphics continue to be updated
Screen logic	On open	Commands and expressions to be evaluated when the screen is opened, similar to a math worksheet
	While open	Commands and expressions to be evaluated when the SCREEN is opened, similar to a math worksheet. If a tag/expression is configured in Trigger , the listed expressions are evaluated once each time the trigger variable changes while the screen is open
	On close	Commands and expressions to be evaluated when the screen is closed, similar to a math worksheet

Screens and Graphics

Pane / Element name		Description
Focus	Receive focus on open	When the screen is opened, the focus automatically goes to the first object in the screen (according to object ID) that can receive focus, as if the screen was selected using the tab key
	Share tab order with other screens	When the last object in the SCREEN is left with the tab key the focus goes to the next open screen (according to Tab order below) rather than back to the first object in the current screen
	Tab order	Similar to object ID for screen objects, this option determines the tab order between screens when multiple screens are open. After tabbing through the last object in a screen, the focus goes to the open screen with the next higher tab order number. Each screen is to have a unique tab order number between 0 and 32767

Background Color or Picture of a WinStudio Screen

A WinStudio screen can have either a solid background color or an editable background picture.

Selecting the background color

By default, a newly created screen has a solid white background.

To change this background color, proceed as follows:

1. Open the screen to be edited in the editor
2. Open the context menu of the screen by right-clicking on a blank spot of the screen. Select the **Background Color** entry. A standard color selection field is displayed as context menu
3. Use the color picker to select a color. The color applies for the entire background of the screen



If a background color is only to be set for a part of a screen, draw a [shape object on page 167](#) and [send it to the background on page 234](#).

Activating the background picture

To use a background graphics for a WinStudio screen and to edit it, proceed as follows:

1. Open the WinStudio screen in the editor
2. Open the context menu of the screen and select the **Screen Attributes** entry. The [Screen Attributes dialog on page 156](#) is displayed
3. Select the **Enable Background** option

A new BMP file with the same name as the WinStudio screen is automatically saved in the "Screens" subfolder of the project folder

(e.g. ..\Screens\<ScreenName>.BMP)

4. Click **OK** to close the "Screen Attributes" dialog
5. In the main menu, open the entry **WinStudio Screen ► Edit ► Background Picture**.

Microsoft Paint is started and the BMP file is opened for editing

To use an image editor other than Microsoft Paint, manually edit "Program Settings.ini" by adding the following setting:

[Options] ImageEditor=filepath

Existing graphics file as background

To select an existing graphics file, especially if it's in a format other than BMP, proceed as follows:

1. Copy the graphics file to be used into the "Screens" subfolder of the project folder

2. Open the respective WinStudio Screen in the development environment
3. Select the entry **Screen Attributes** in the context menu of the screen. The "Screen Attributes" dialog is displayed
4. Select the **Enable Background** option
5. Select the **Shared Image** option and enter the name of the graphics file in the field (without file extension)
6. Select the graphics format of the graphics file from the list



In projects for the target system with Windows CE, only BMP files are supported

7. Click **OK** to close the "Screen Attributes" dialog. If the development application can find and load the specified graphics file, it is displayed in the WinStudio screen. If it cannot be opened, a warning message is displayed

8.2.2 Screen Groups Folder

The "Screen Groups" folder contains elements that combine several [screens on page 156](#) to groups.

To open a specific screen group, open the "Screen Group" folder and double-click on the element.

To remove a specific screen group, right-click on the respective element and select "Delete" in the context menu.

To create a new screen group, proceed as follows:

1. Open the context menu of the "Screen Groups" folder and select "New screen group". The configuration dialog for screen groups is displayed:

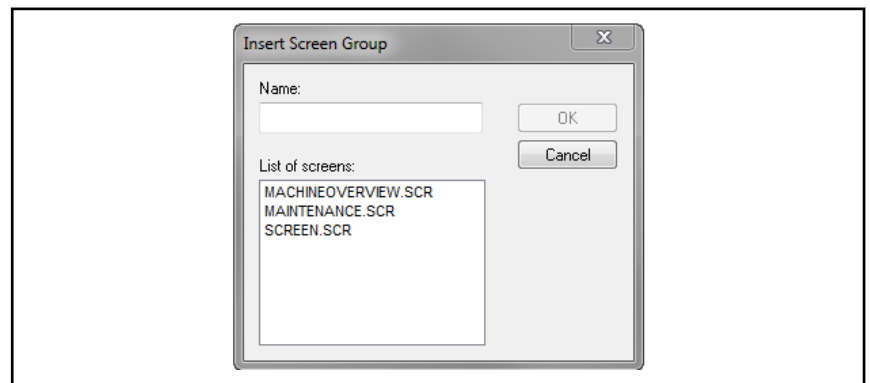


Fig.8-4: Insert Screen Group dialog

2. Enter a name for the new screen group into the **Name** field.
3. Select the screens from the list that are to be included in the screen group.

The list only contains current screens in the "WinStudio" folder or a screen subfolder.

To select multiple screens, press the <Ctrl> key while clicking on the screen names.

Release the <Ctrl> key when all screens are selected.

4. Click **OK** to close the "Insert screen group" dialog.

8.2.3 Web Pages Folder

"WinStudio" can also save screens in HTML format to display them in a web browser.

HTML pages cannot be created directly. They are created from already existing WinStudio screens and stored in the "Web folder".

To create an HTML page, a WinStudio screen has to be created first. Configure a screen as usual (create objects, add properties, etc.) and observe that this screen is to be used on as a web page. After finishing, save the screen as usual. Afterward, select "Save as HTML" in the context menu of the WinStudio screen in the Project Explorer.

NOTICE

The Web pages that are created by clicking "Save as HTML" are independent of the screen files from which they were generated.

Consequently, if the WinStudio screen is modified, the changes do not appear on the web page until "Save as HTML" is selected again.

To view the web pages during runtime, the project settings have to be configured first.

For this, select the entry **Tools ► Web Settings** in the context menu of the "WinStudio" folder.

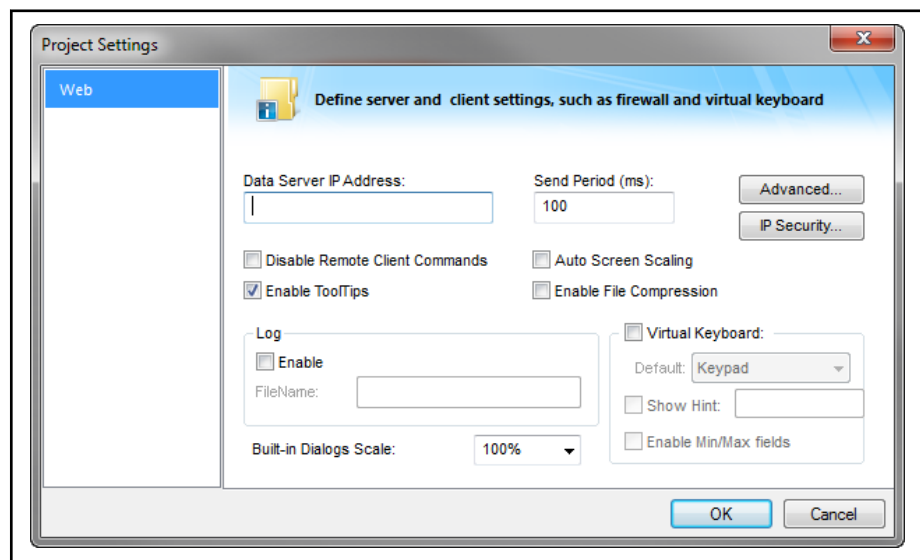


Fig. 8-5: Project Settings dialog - Web Tab

1. Enter the IP address of the computer on which the project is executed into the **Data Server IP Address** field
2. Enter a value into the **Send period (ms)** field to specify the send period (in milliseconds) used to exchange data between the server and the Thin Client stations
3. Click (enable) the following checkboxes if applicable:
 - **Disable Remote Client Commands:** Enable this option to prevent a remote client from issuing commands from the Thin Client to the server
 - **Enable ToolTips:** Enable this option to display Windows ToolTips when displaying WinStudio screens on the Thin Client (browser)

Screens and Graphics

- **Auto Screen Scaling:** Enable this option to automatically adjust the size of the WinStudio screens in a browser window. Enable this option for remote execution on a Thin Client if the screen size is to be adjusted automatically by "WinStudio" if the size of the browser window is changed
 - **Enable File Compression:** Enable this option to compress the files stored in the "Web" subfolder of the project folder. This option is useful for reducing download time, particularly for slow connections between server and Thin Client
4. Click the "IP Security" button to open the IP Security dialog (see figure)
Use the parameters of this dialog to specify the range of IP addresses for the computers that are allowed to access the project as Thin Clients

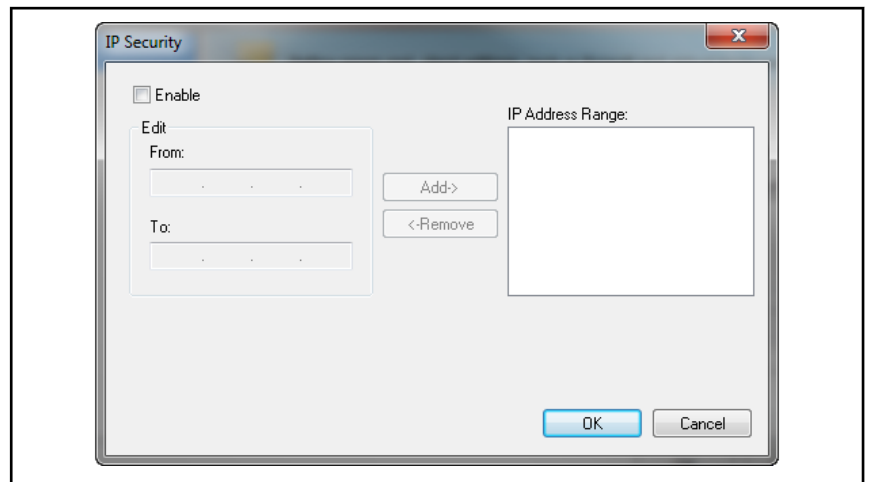


Fig. 8-6: IP Security dialog

Enable the option "Enable" to specify one or more IP address ranges to connect via a web browser

Afterwards, enter the bottom limit of the IP address range in the "From" field and the upper limit into the "To" field

Use the "Add" and "Remove" buttons to move the IP addresses into the "IP Address Range" list. "WinStudio" permits the computers listed in this pane to access the project as Thin Clients

5. To enable logging for the Thin Client, click (enable) the "Enable" check box in the "Log" pane and enter a file name into the "Filename" field to generate a log file on the Thin Client station. This log file can be used for debugging purposes
6. Click **OK** to close the "Project Settings" dialog



If any of the web information in the "Project Settings" dialog is changed, the project has to be re-verified for the new setting to take effect. To verify the project, select the "Verify Application" entry in the context menu of the "WinStudio" folder. (If any windows are open in the development system, "WinStudio" asks to close them before verifying the project).

As the web pages display data from the project through the web server, runtime system, web server and TCP/IP server have to be executed to view the web pages.

For further information on WinStudio web applications, refer to [Provision as a Web Application on page 391](#).

Screens and Graphics

8.3 Using Screen Objects and Animations

8.3.1 Editing Screen Objects

Editing Screen Objects, General Information

To edit WinStudio screens, the toolbars offer numerous functions in addition to the functions in the context menu of the screen editor.

To display or hide toolbars, right-click on the tool bar area and enable or disable the required toolbars.

All functions available in the WinStudio toolbars can also be found in the IndraWorks main menu under the "WinStudio Screen" menu item.

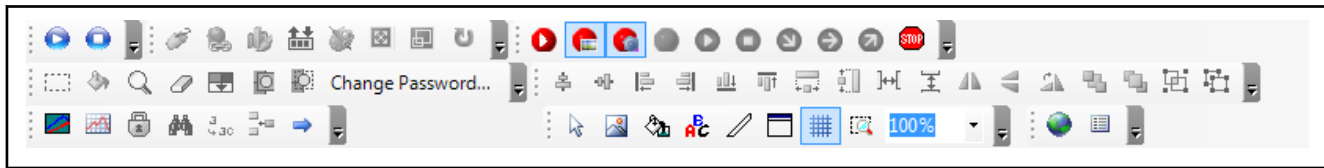


Fig. 8-7: WinStudio Toolbars

Editing Object Properties

The Object Properties dialog displays the configurable properties of a screen object or animation. Each type of screen object and animation has its specific properties, but all types are displayed in the same basic dialog.

Opening the dialog To open the "Object Properties" dialog, select one of the following options:

- Select the screen object and press <Alt>+<Enter>
- Right-click the screen object and click **Properties** in the context menu
- Double-click on the screen object.

Dialog in detail All objects have the following properties in common:

	Description
Pin needle 	If the pin needle is pressed, the "Object properties" window remains opened as long as the screen editor is open. In this mode, the properties of all screen objects can be subsequently displayed or edited without having to close and reopen the dialog
Replace	Opens the "Replace" dialog to replace strings, tags or properties on page 164 for the selected object or group of objects
Hint	If the cursor is moved over the object during runtime, a tooltip is displayed. This can be used to provide quick support The text in the Hint field is also written temporarily to the System tag on page 110 "Hint" so that an action according to this tag value can be triggered by moving the cursor over a specific object. To display Hints/ToolTips during runtime, select the option Enable ToolTips in the "Project settings" dialog. This function can be activated/deactivated separately for the standard viewer (on the Viewer tab of the WinStudio Project Settings) and for Web Thin Clients (in the Web settings dialog)
	By means of the combo box at the right of the dialog, specific objects within a group of objects or animations can be selected for editing

Replacing Project Tags in a Document or Screen Object

To replace all tags in a screen object, double-click the object to open its "Object Properties" dialog and then click **Replace**.

To replace all tags in the current document, carry out one of the two following actions:

Screens and Graphics

- Open the "WinStudio Screen" menu in the IndraWorks main menu and select the entry **Replace tags and strings in the current document**
- Display the "WinStudio Table Functions" toolbar and click on the **Replace tags and strings in the current document** button.

In both cases, the "Replace" dialog is displayed as described below.

One or more tags can be replaced by clicking the **Whole Tag Name** tab. The left column, **From**, displays the tags currently used. Into the right column, **To**, the new names can be entered.

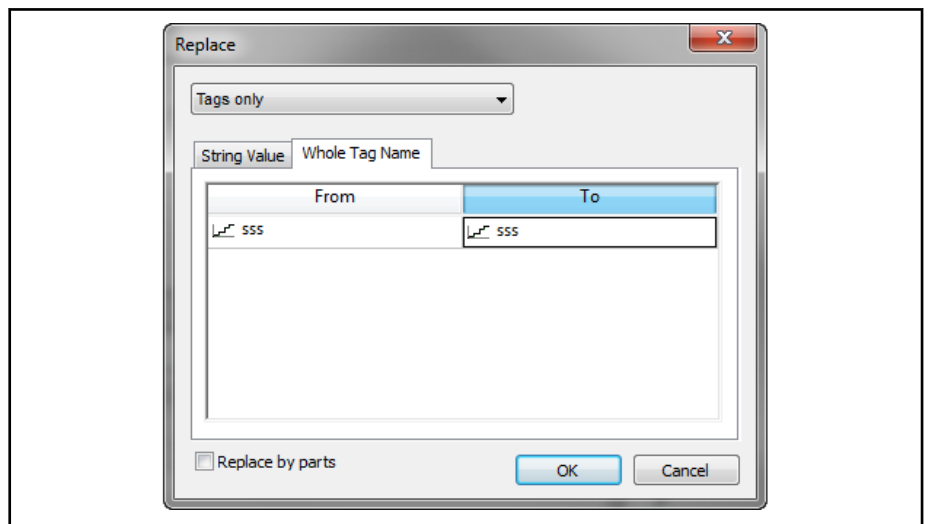


Fig. 8-8: "Replace" dialog, "Whole Tag Name" tab

Note that this does not "rename" or "delete" any tag. Tags used in the object are only replaced by other tags from the database.

By activating the "Replace by parts" option, replacing parameters (e.g. array index) within classes and arrays becomes easier.

One or more strings can also be replaced (e.g., button captions, descriptive texts) by clicking the **String Value** tab.

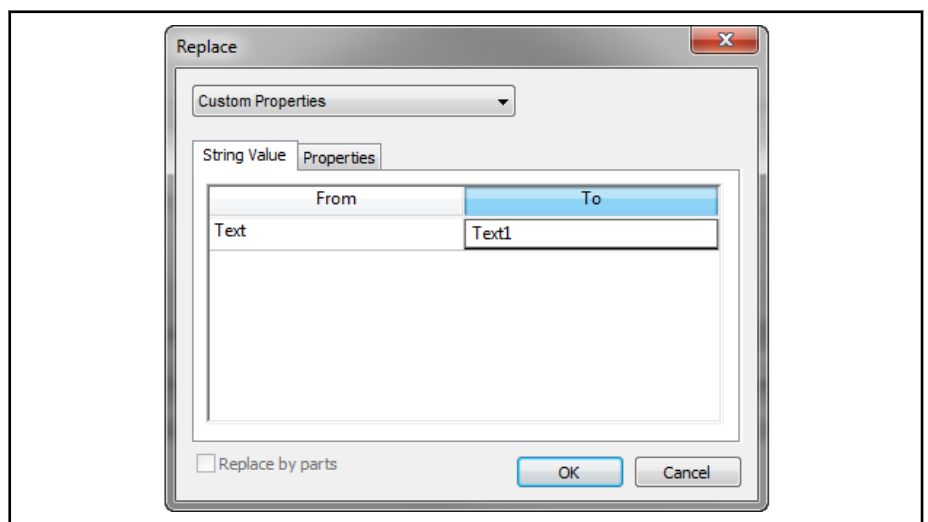


Fig. 8-9: "Replace" dialog, "String Value" tab

After editing all entries, click **OK**.

Screens and Graphics

Arranging Objects

The "WinStudio Alignment Functions" and "WinStudio Object Properties" toolbars offer numerous functions to align, group and design screen objects.



Fig.8-10: "Alignment Functions" toolbar

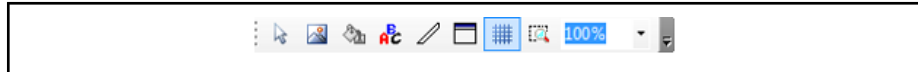


Fig.8-11: "Object Properties" toolbar



Some functions in the toolbars are only available if one or multiple objects have been selected in the WinStudio screen.

Move the cursor over an active "tool" to display the function in plain text.

The tools are organized into the following groups:

- **Arrange:** Arranging objects in a WinStudio screen including
 - [Move to Foreground and Move to Background on page 234](#)
 - [To the Foreground and To the Background, by One Level on page 235](#)
 - [Group and Ungroup Objects on page 235](#)
 - [Align, Center and Distribute on page 236](#)
 - [Rotating and Mirroring on page 238](#)
- [Adjusting the Position on page 239](#)
- **Size:** [Same Width / Same Height for Selected Objects on page 239](#) of a screen object in a WinStudio screen
- **Style:** Changing the color of screen objects
 - [Fill Color on page 240](#) of a screen object
 - [Line Color on page 241](#) of a screen object
- **Edit the color of a component:** [Color Interface on page 241](#)
- **Fonts:** Changing the [Fonts on page 244](#) of a screen object

Locking Objects in the Screen Editor

The function for moving objects in the screen editor can be locked to prevent accidental position changes of screen objects after setting the final screen layout.

- In the WinStudio screen editor, open the context menu and select the entry **Lock moving**
- To allow moving screen objects, execute the function **Lock moving** again

Optionally, the "Lock moving" function can be activated and deactivated by pressing <Ctrl>+<D>.

Undo

In the WinStudio screen editor, the last action can be undone by selecting **Undo** in the "Standard" toolbar.

Optionally, press <Ctrl>+<Z>. This way, up to 20 actions before the last action can be undone.

(Actions in the "WinStudio Object Properties" toolbar are not included in the **Undo** count, i.e. adjusting these settings cannot be undone with this function.)

Grid Settings

To display/hide the grid in the screen editor, click on **Activate/deactivate grid** in the "WinStudio Object Properties" toolbar.

To edit the grid settings, open the "Grid" dialog. Right-click in a free area in the screen editor and click **Grid Settings** in the context menu.

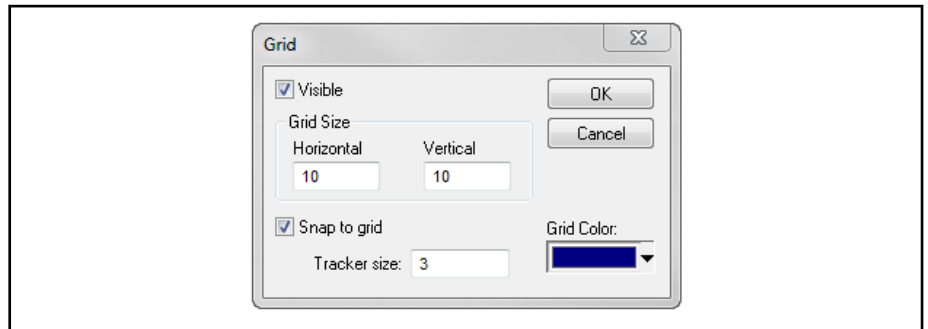


Fig. 8-12: Grid settings dialog

8.3.2 Static Objects

Static Objects, General Information

After opening the WinStudio editor, the "WinStudio Screen Tools" tool window becomes available.

The tool window includes various objects that can be inserted into a WinStudio screen: Rectangles, polygons, lines, text fields, etc.

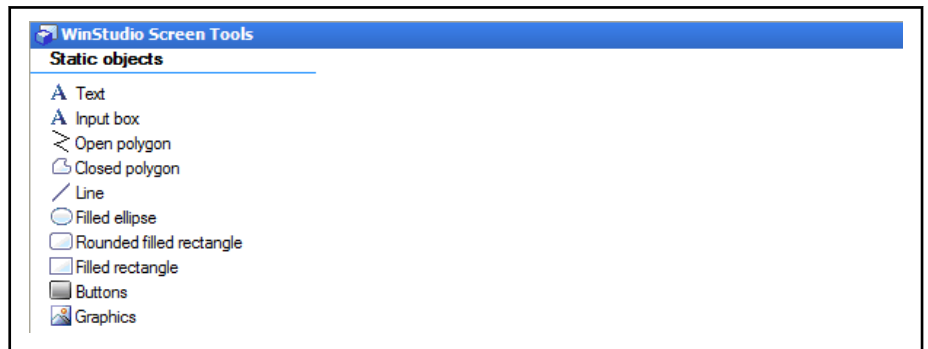


Fig. 8-13: "WinStudio Screen Tools" tool window

Line Object

To draw an orthogonal line, click on the **Line** entry in the tool window.

Afterwards, click into the drawing area of the WinStudio screen editor:

1. Left-click to set the starting point of the line
2. Drag the line with the cursor to adjust the line size
3. To align the line, drag at one of the marked end points
4. To display the object properties, double-click on the object

The "Object Properties" dialog is displayed as follows:

Screens and Graphics

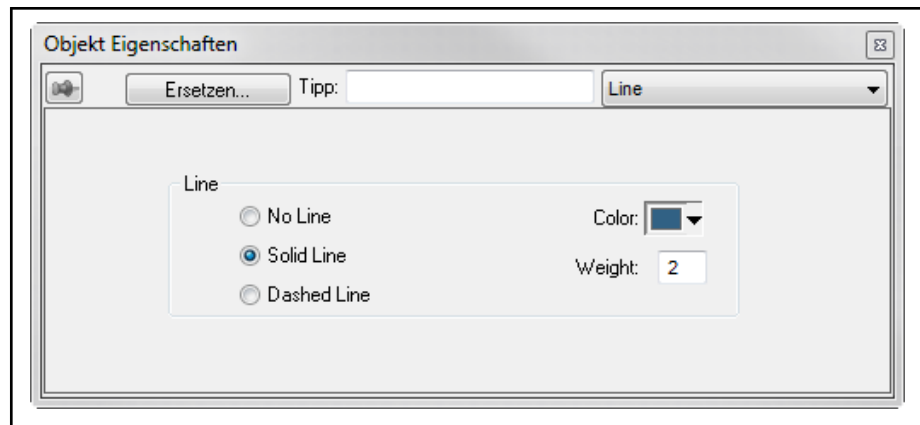


Fig. 8-14: Object Properties: Line

Use the "Object Properties" dialog to specify the following parameters for the line:

- **Line:** Specify a line style by clicking the **No Line**, **Solid Line** or **Dashed Line** button
- **Color:** Specify a line color by clicking the **Color** button. In the "Color" dialog, click on a color to select it and close the dialog
- **Weight:** Specify the line width (in pixels) by entering a number representing the line width into the text field

Open Polygon Object

In the **WinStudio Screen Tools** tool window, click on **Open polygon** to draw an open polygon with frame in the specified foreground color.

To draw an open polygon in the drawing area, proceed as follows:

1. Left-click to set the starting point of the polygon
2. Move the cursor to a new location and click again to place the second point
3. Repeat this process until the desired polygon is created
4. Double-click to stop drawing the polygon

To change the shape of a polygon after drawing it, select it and drag any of its points



If individual points of a polygon cannot be dragged, they may be [grouped on page 235](#). To ungroup the points, right-click on the polygon and choose **Ungroup** from the context menu

To display the object properties, double-click on the object. The "Object Properties" dialog is displayed as follows.

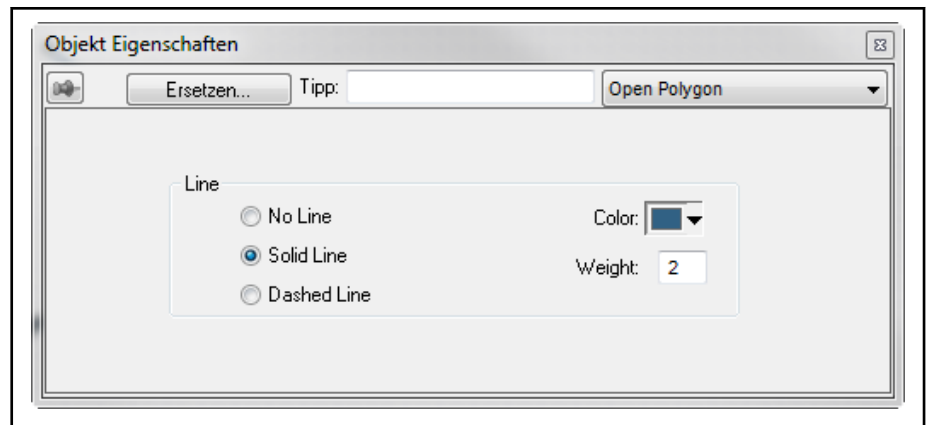


Fig. 8-15: Object Properties: Open Polygon

Use the "Object Properties" dialog to specify the following parameters for the polygon:

- **Line:** Specify a border line style by clicking the **No Line**, **Solid Line**, or **Dashed Line** button
- **Color:** Specify a border line color by clicking the **Color** button. In the Color dialog, click on a color to select it and close the dialog
- **Weight:** Specify the border line width (in pixels) by entering a number specifying the line width into the text field

Closed Polygon Object

In the **WinStudio Screen Tools** tool window, click on **Closed polygon** to draw a closed polygon with frame in the specified foreground color.

To draw a closed polygon in the drawing area, proceed as follows:

1. Left-click to set the starting point of the polygon
2. Move the cursor to a new location and click again to place the second point
3. Repeat this process until the desired polygon is created
4. Double-click or right-click to stop drawing the polygon
5. To display the object properties, double-click on the polygon object

To change the shape of a polygon after drawing it, select it and drag any of its points



If individual points of a polygon cannot be dragged, they may be [grouped on page 235](#). To ungroup the points, right-click on the polygon and choose **Ungroup** from the context menu

The "Object Properties" dialog is displayed as follows.

Screens and Graphics

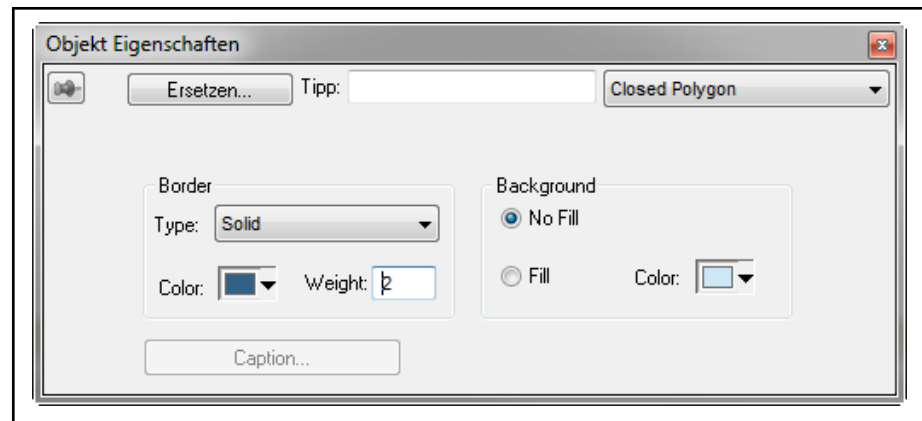


Fig. 8-16: Object Properties dialog: Closed Polygon

Use the "Object Properties" dialog to specify the following parameters for the polygon:

- **Type:** Specify a border line style by clicking the **No Line**, **Solid Line**, or **Dashed Line** button
- **Color:** Specify a border line color by clicking the **Color** button. In the "Color" dialog, click on a color to select it and close the dialog
- **Weight:** Specify the border line width (in pixels) by entering a number specifying the line width into the text field
- **Fill effect:** To specify whether the polygon is filled, click **No Fill** or **Fill**

If the **Fill** option is selected, choose a fill color by clicking on the **Color** button. In the "Color" dialog, click on a color to select it and close the dialog

Rectangle Object

To create rectangles, click on **Filled rectangle** in the **WinStudio Screen Tools** tool window and proceed as follows:

1. Click in the drawing area and drag the mouse/cursor to draw the rectangle
2. Release the mouse button when the rectangle has the desired size
3. Double-click on the object to open the "Object Properties" dialog.

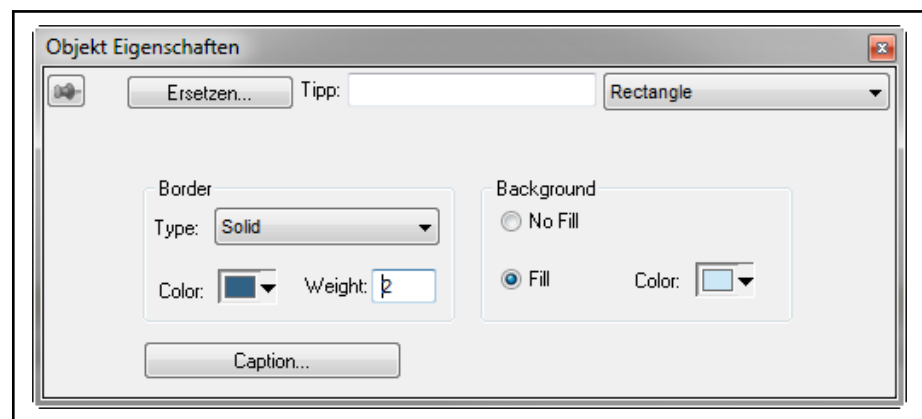


Fig. 8-17: Object Properties: Rectangle

Use the "Object Properties" dialog to specify the following parameters for the rectangle:

- **Type:** Specify a border line style by selecting **None**, **Solid**, **Dashed**, **Etched**, **Raised** or **Sunken**

Screens and Graphics

- **Color:** Specify the border line color by clicking the **Color** button to open the "Color" dialog. Select a color and close the dialog
- **Weight:** Specify the border line width (in pixels) by entering a number into the respective text field
- **Background:** To specify the background color of the rectangle, select **No Fill** or **Fill**
If the **Fill** option is selected, specify a fill color by clicking on the color rectangle. If the Color dialog is displayed, select a color and close the dialog
- **Caption:** Click on this button to open the "Caption" dialog and to enter a display text

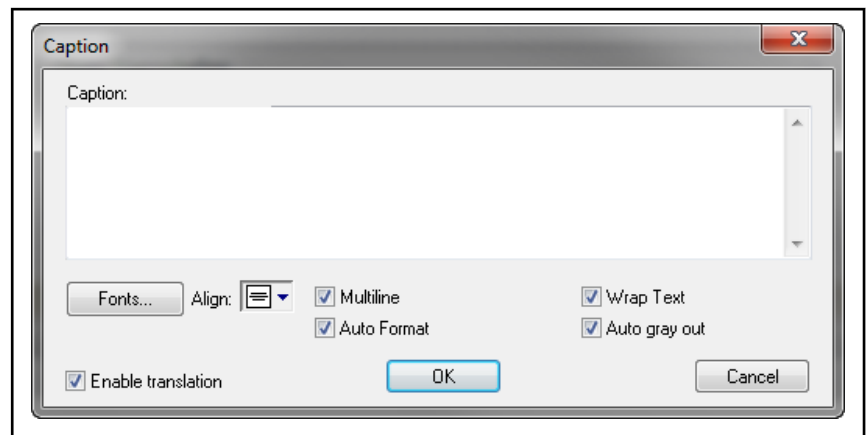


Fig. 8-18: Caption dialog

- **Caption:** Enter the text to be displayed inside the rectangle object. Tags can be included in curly brackets (e.g. {tagname})
- **Fonts:** Specify a font style for the text by clicking the **Fonts** button
- **Align:** Specify the alignment for the caption of the rectangle
- **Multiline:** Activate this option to display the caption of the rectangle in more than one line
- **Auto Format:** If this option is enabled and if the caption includes a decimal value enclosed by curly brackets (e.g. {1.2345}) or a tag of [Real on page 113](#) type (see **Caption** above), the value is formatted according to the virtual table created by the function [SetDecimal-Points on page 604](#)
- **Wrap Text:** If enabled, the object automatically wraps the text when necessary
- **Auto gray out:** The caption is grayed out at runtime if the [command animation on page 220](#) applied to the rectangle is disabled by means of the **Disable** field or due to the security system
- **Enable translation:** If this option is enabled (activated), translation is enabled during runtime by means of the [translation tool on page 381](#)

Rounded Rectangle Object

To draw rounded rectangles, click on **Rounded Rectangle** in the **WinStudio Screen Tools** tool window and proceed as follows:

1. Click in the drawing area and drag the mouse/cursor to create the rectangle
2. Release the mouse button to stop drawing the object

Screens and Graphics

3. Double-click on the object to open the "Object Properties" dialog.

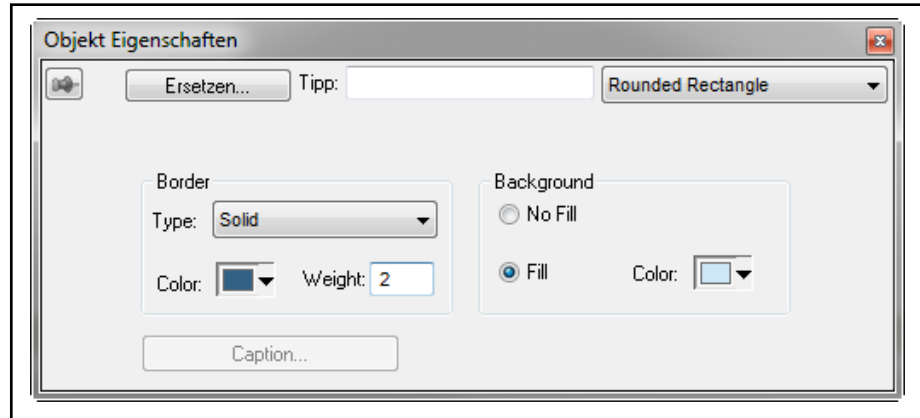


Fig.8-19: Object Properties: Rounded rectangle



The "rounded rectangle" element cannot be used to create bar graphs for projects running on Windows CE target systems.



Rounded rectangles have one extra handle in the bottom-right corner to modify the arc angle.

Use the "Object Properties" dialog to specify the following parameters for the rectangle:

- **Line:** Specify a border line style by clicking the **No Line**, **Solid Line**, or **Dashed Line** button
- **Color:** Specify the border line color by clicking the **Color** button to open the "Color" dialog. Click the color to select it and close the dialog
- **Weight:** Specify a line width by entering a number (in pixels) into the text field provided
- **Background:** Specify the background color of the rectangle by clicking **No Fill** or **Fill**. If the **Fill** option is selected, choose a fill color by clicking on the **Color** button
In the "Color" dialog, click on a color to select it and close the dialog
- **Caption:** This option is not enabled for this object

Ellipse Object

To draw ellipses, chords, arcs and rings, click on **Ellipse** in the **WinStudio Screen Tools** tool window:

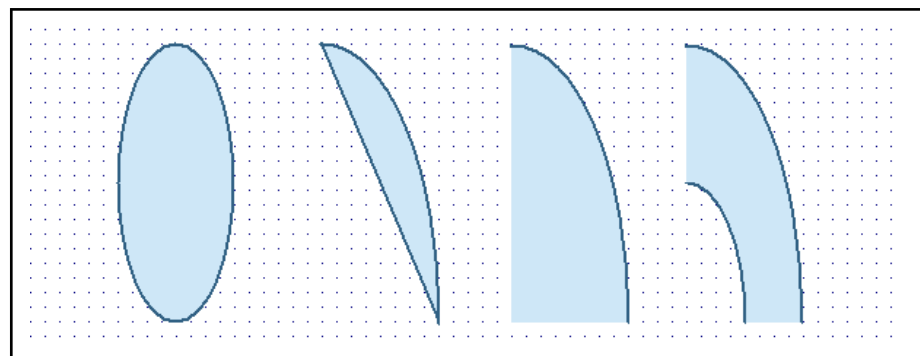


Fig.8-20: Ellipse, chord, arc and ring



The "Ring" style is particularly useful to create installation drawings.

To create an ellipse, proceed as follows:

1. Click in the drawing area and drag the mouse/cursor to create an ellipse shape
2. Release the mouse button to stop drawing the ellipse
3. Use the "Object Properties" dialog to change the shape to a chord, arc or ring
4. Double-click on the object to open the "Object Properties" dialog.

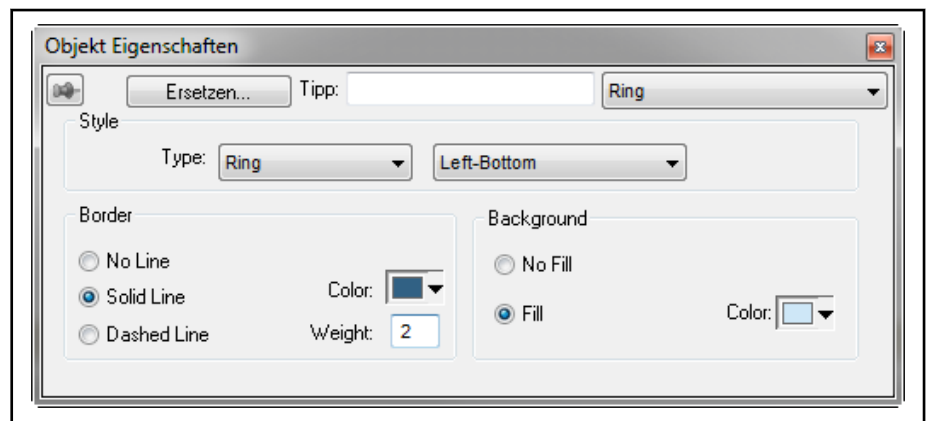


Fig. 8-21: Object Properties: Ellipse

Use the "Object Properties" dialog to specify the following parameters for the ellipse:

- **Style:** Specify the object style by selecting **Ellipse**, **Arc**, **Chord**, or **Ring** from the drop-down list
For the objects "arc, chord or ring", additionally specify the quadrant for illustration (bottom left, top left, bottom right, top right)
For example, to represent a half-circle pipe, create two **ring** objects. Specify one as **Left-Bottom** and the other as **Right-Bottom** then join the two objects to create a half-pipe
- **Background:** To specify the background color of the ellipse, select **No Fill** or **Fill**
If the **Fill** option is selected, choose a fill color by clicking on the **Color** button
In the "Color" dialog, click on a color to select it and close the dialog
- **Border:** Specify a border line style for the ellipse by selecting **No Line**, **Solid Line** or **Dashed Line**
- **Weight:** Specify a line width by entering a number (in pixels) into the text field provided

Adding Graphics to a WinStudio Screen

To add graphics to WinStudio screens, copy the screen to the clipboard and paste it directly into the screen editor.

This task assumes that a WinStudio screen has been opened in the editor.

Screens and Graphics



Please note that using this method to add a graphic to the screen the graphics data is embedded in the WinStudio screen file.

With a link to an external graphics file, the screen, however, can be changed during runtime. To do so, the screen object [Linked Picture on page 219](#) is used.

1. Open the desired graphic in an image editing application, such as Microsoft Paint
2. Select part or all of the graphic, either by using the **Select** tool or by pressing <Ctrl>+<A>
3. Copy the selected graphic to the clipboard, either by using the **Copy** tool or by pressing <Ctrl>+<C>
4. Switch to the "WinStudio" screen editor
5. Open the context menu of the open WinStudio screen editor and select **Paste** or press <Ctrl>+<V>. The selected graphic is inserted into the WinStudio screen as bitmap object
6. Double-click on the screen object. The "Object Properties: Bitmap" dialog is displayed

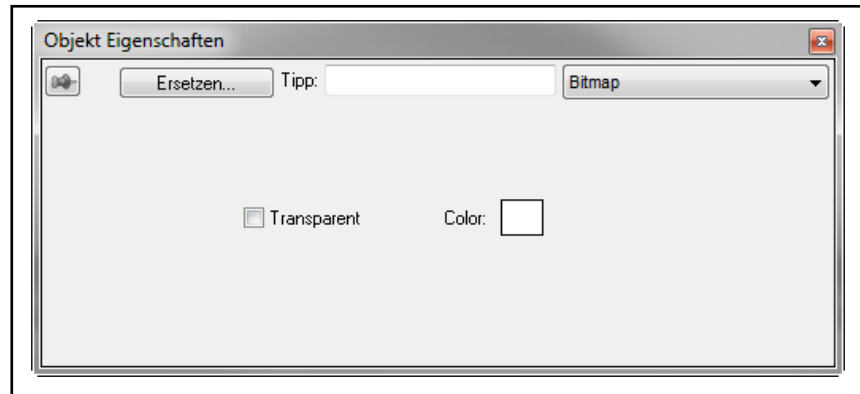


Fig.8-22: Object Properties: Bitmap

7. If the background of the inserted screen is to be transparent, select **Transparent** and use the color picker to specify the transparent color
8. Close the "Object Properties" dialog

8.3.3 Active Objects

Active Objects, General Information

The "WinStudio Screen Tools" tool window offers the following control elements to create interactive objects.

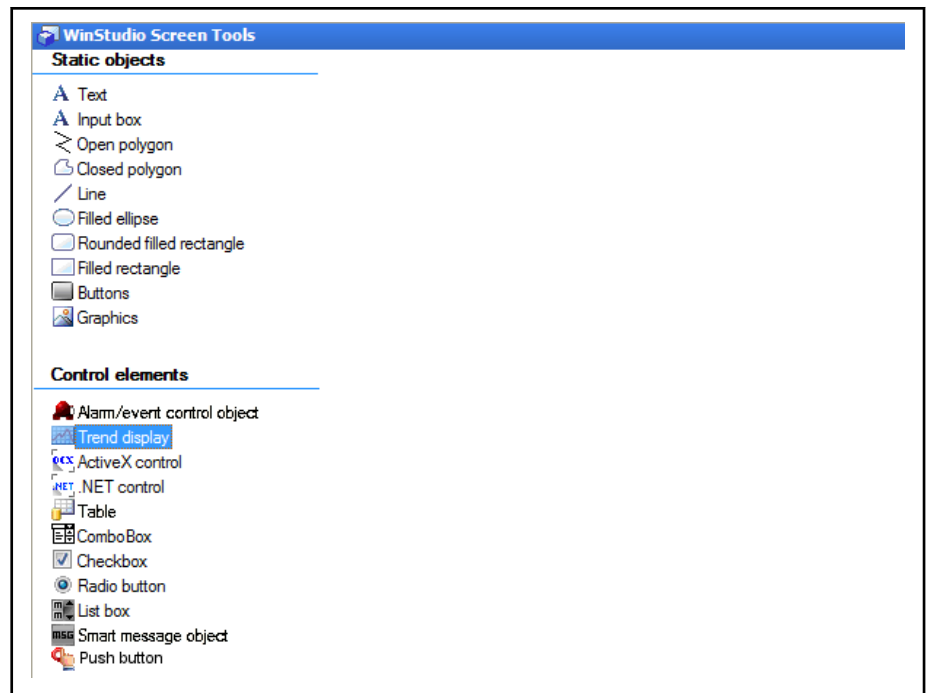


Fig. 8-23: WinStudio Screen Tools

Text Object

To create text objects, click on **Text** in the **WinStudio Screen Tools** tool window and proceed as follows:

1. Click in the drawing area
Enter a line of text when the cursor is displayed
2. After entering the string, double-click on the new text object to view the "Object Properties" dialog

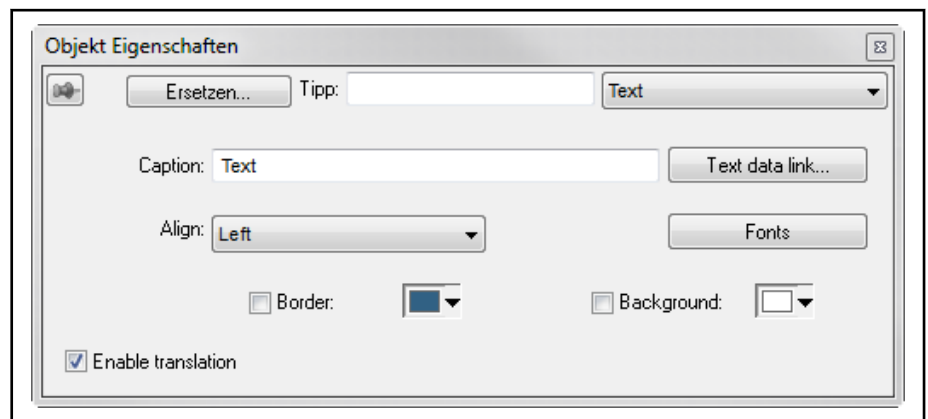


Fig. 8-24: Object Properties: Text

Use the "Object Properties" dialog to specify the following properties:

- **Caption:** Input field to edit the string entered in the "Text" object
- **Text data link** button: Click to apply the [text data link animation on page 225](#) to the "text" object. If the caption does not include any placeholder characters ("###") for the text data link, click this button to automatically add those characters
- **Align:** Align the text by selecting **Left**, **Center** or **Right** from the combo box

Screens and Graphics

- **Fonts:** Specify the following text parameters by clicking the **Fonts** button:
 - **Fonts**
 - **Font style**
 - **Size**
 - **Effects**
 - **Color**
 - **Script**
- **Border:** Activate the "Border" checkbox to add a border to the text field and click on the color selection button to the right to select a suitable border color
- **Background:** Specify a background color by clicking on **Background**
In the "Color" dialog, click on a color to select it and close the dialog
- **Enable translation:** Enable this option to apply the text to the translation file

Button Object

To insert buttons, click on **Button** in the **WinStudio Screen Tools** tool window and proceed as follows:

1. Click in the drawing area and drag the mouse/cursor to create the button
2. Release the mouse button when the button has the desired size
3. Double-click on the object to open the "Object Properties" dialog.

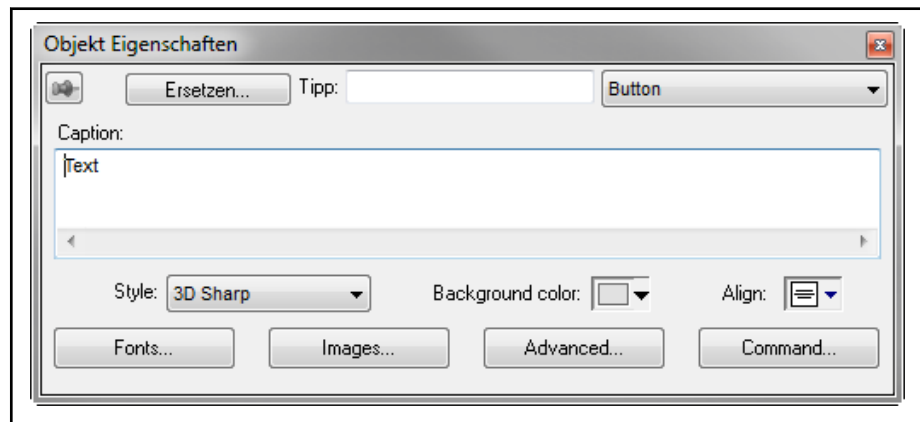


Fig. 8-25: Object Properties: Button

The following parameters of "Button" objects can be configured in the "Object Properties" dialog:

- **Caption:** Text input field for the button caption. The display text may include tags. These have to be entered between curly brackets (e.g. {tagname})
- **Style:** Select one of the following display types for buttons:
 - **3D Sharp:** A raised button with visible contour at the border, suitable for touchscreens
 - **3D Soft:** A raised, rounded button suitable for touchscreens
 - **OS Like:** A button styled to match the operating system on which the project client is running, suitable for Windows desktops with Web Thin Client or Secure Viewer

- **Standard:** Flat standard button

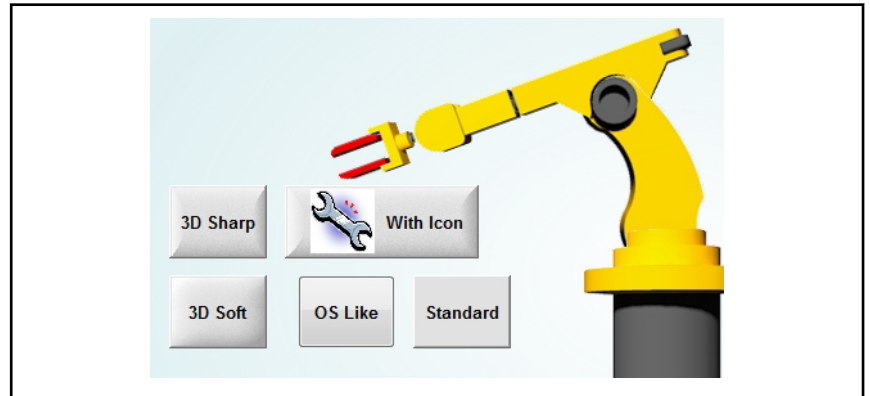


Fig. 8-26: Examples of buttons

- **Background color:** Select a background color for the button
- **Align:** Select the alignment for the caption of the button
- **Fonts:** Set the following caption parameters by clicking the **Fonts** button:
 - Fonts
 - Font style
 - Size
 - Effects
 - Color
- **Images:** Click on **Images** to insert an image file
Specify the following parameters in the "Images" dialog:
 - **File:** Enter the file path to the graphics file. To open a standard Windows file browser, click on the "Search" button next to the box
 - By default, the image is displayed at its actual size. To change this, select User-defined from the Size list and configure the desired image width and height (in pixels)
 - **Position:** Select the position of the graphic in relation to the caption
 - **Offset:** Specify the offset (in pixels)
 - **Transparent Color:** Select which color in the image is to be transparent. The background color of the button (see above) is visible in these areas
- **Advanced:** Specify advanced settings for the button by clicking **Advanced**
In this dialog, the following parameters can be set:
 - **Enable translation (optional):** Enable this option to apply the text to the translation file
 - **Multiline:** If this option is enabled, the caption of the button can be displayed in more than one line
 - **Wrap Text:** If this option is enabled, the object automatically wraps the text when necessary
 - **Auto gray out:** The caption is grayed out at runtime if the [Command animation on page 220](#) applied to the button is disabled by means of the "Disable" field or due to the [security system on page 357](#)
 - **Auto Format:** If this option is enabled and if the caption includes a decimal value enclosed by curly brackets (e.g. {1.2345}) or a tag of

Screens and Graphics

[Real on page 113](#) type, the value is formatted according to the virtual table created by the function [SetDecimalPoints on page Set-DecimalPoints](#)

- **Command:** Click to automatically apply a [command animation on page 220](#) to the button and switch to the animation properties

Pushbutton Object

Select **Pushbutton** in the **WinStudio Screen Tools** tool window to create a pushbutton object by means of the [command animation on page 220](#).

"WinStudio" offers the following preconfigured types of pushbuttons. The button types show the behavior of the hardware buttons with the respective name:

- **Momentary** (default): Changes the state (**Open** or **Closed**) if pressed and returns to its initial state if released. This pushbutton type is always displayed in its normal position when the WinStudio screen is opened
- **Switch:** Changes the state (**Open** or **Closed**) if pressed but does not return to its initial state if released. Press the button again to change its present state. This pushbutton type retains its state if the WinStudio screen is changed
- **Locked:** Changes the state (**Open** or **Closed**) if pressed and remains in this state until it is released by changing the "Reset" tag

"WinStudio" provides the following button styles:

- Button with frame and control LED
- Button without frame (filled rectangle) as default

In 3D or with shading.

To add one or more preconfigured buttons to a screen, proceed as follows:

1. Click the **Pushbutton** tool, and position the mouse (pointer) in the WinStudio screen editor
2. Create the rectangular pushbutton by click and draw
Release the mouse button when the button has the desired size
3. Double-click on the object to open the "Object Properties" dialog.



Alternatively, open the "Object Properties" dialog by right-clicking on the pushbutton object or by selecting the object and pressing <Alt>+<Enter> and selecting **Properties** from the context menu

Object properties: Pushbutton

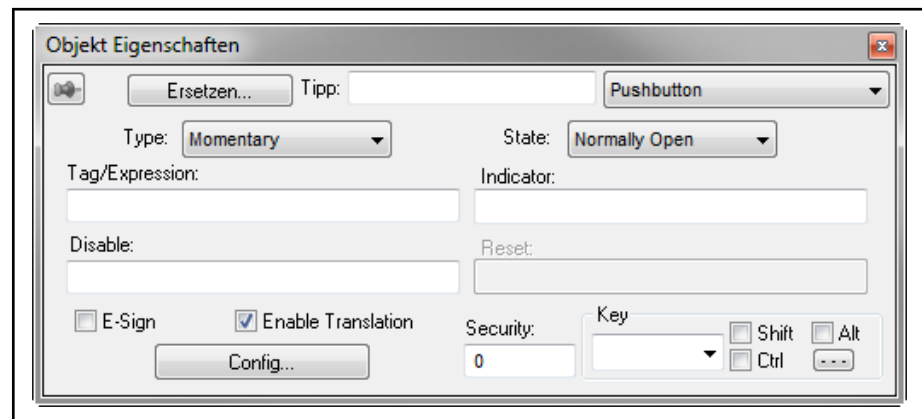


Fig. 8-27: Object Properties: Pushbutton

Use this dialog to specify the following parameters:

- **Type:** Click to select the pushbutton type (**Momentary** (default), **Switch** or **Locked**)
- **Status:** Select the button function logic by clicking on the "State" selection field. The button can be projected as N/O (normally Open) or N/C (normally Closed) contact
For example: The "Momentary" button is projected as "N/O contact" (normally Open). As long as the button is not actuated, the runtime system reports "open" for the button. The same button projected as "N/C contact" (normally closed) reports "closed" at runtime without actuation
- **Tag/Expression:** Input field to select a WinStudio tag or to enter a program expression:
 - Enter a tag to which a certain value is assigned at runtime. This value depends on the state of the button (e.g. pressed), the pre-configured logics (e.g. N/O) and the "write value" parameterized in the "Configuration" menu
 - Enter a program expression to be executed on pressing the button



"WinStudio""does not" write the result of any expression in the **Tag/Expression** field into a tag

- **Indicator:** Enter a tag to define an indicator that causes the pushbutton to change to a specified color if the tag value matches one of two specified values. Both the colors and tag values have to be defined in the "Configuration" dialog. If this field is left blank, the indicator changes color automatically on pressing the button
- **E-Sign:** If this option is enabled, a prompt is displayed to enter the electronic signature before executing the animation
- **Reset** (only active for pushbuttons of the **locked** type): Enter a tag to control the locking state of the pushbutton as follows:
 - Enter a zero and the pushbutton remains in a locked state after pressing it
 - Enter a non-zero value and a latched pushbutton becomes unlatched after pressing it. The tag value has to be reset to zero before the pushbutton can be pressed again
- **Key** pane: Specify a keyboard code or create a key combination to change the state of a pushbutton if no pointing device (mouse or touchscreen) is available or if shortcuts are to be created in addition to pushbuttons
- **Key** selection list: Enter a key in the text field or select a alphanumeric key from the selection list. Enter only a single character or key. No numbers can be entered in this field. Click (enable) the **Shift**, **Ctrl**, or **Alt** field to create a combination key, meaning the Shift, Ctrl or Alt key has to be pressed with the key specified in the selection list. Click the browse button ... to open the "Key Modifier" dialog to modify the combination keys. Choose **Left**, **Right** or **Left or Right** to specify the position on the [key-board on page 73](#) of the Shift, Ctrl, or Alt combination key. If **Left or Right** is selected, the command is executed any time one of these keys is pressed in combination with the key specified in the selection list
- **Disable:** Enter a non-zero value to disable the pushbutton, thus, pressing the button has no effect. This field is empty by default

Screens and Graphics

- **Enable Translation:** If this option is enabled, the text is applied to the translation file. (Refer to [translation tool on page 381](#) for further information.)
- **Security:** Enter a value to specify a security level (0 to 255) for this button. If the user does not have the specified security level, the pushbutton becomes inactive. If the user has the appropriate security level or this field is left blank, the button is active
- **Config...:** Click to open the "Configuration" dialog to specify "Style" and state parameters for the pushbutton

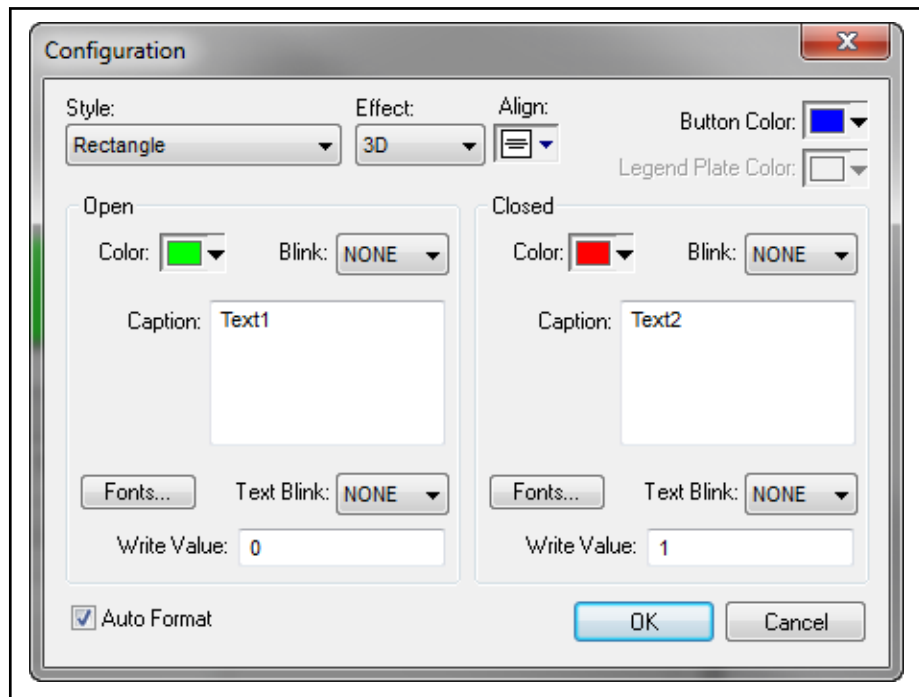


Fig.8-28: Configuration dialog

This dialog provides the following parameters:

- **Style:** Click the combo box button to select a pushbutton style (**Rectangle** (default) or **Rectangle with faceplate**)
- **Effect** combo box: Click to select a 3-D effect for the pushbutton
 - **Floating** (default): Pushbutton resembling a flat object with a shadow
 - **3D:** Pushbuttons having rounded corners and appearing to be "pressed into the screen" when pressed

The **Style** and **Effect** parameters can be combined to create four different pushbuttons, as illustrated in the following figure:

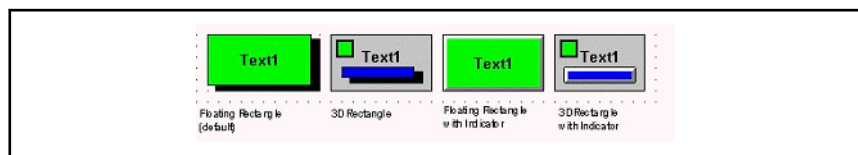


Fig.8-29: Pushbutton styles

- **Align:** Specify the alignment for the caption of the pushbutton
- **Button Color:** Click to define the constant color of the button if a button style "with frame" is selected

Screens and Graphics

For button objects "without frame", the set button color is not displayed or in intervals if the "Blink" option is selected

- **Legend Plate Color:** Click to define the frame color for button styles "with frame" (button background = legend)

This field is disabled if a "button style without frame" has been selected

- **Open and Closed panes:** By means of the following parameters, the appearance of pushbutton objects in open and closed state is configured

- **Color:** Click to specify a default color for an indicator in each **state**. The color switch is only effective for button styles "without frame"
- **Blink:** Click to specify whether the color defined in the **Color** field blinks and how fast it blinks for each state (**None** (no blinking, "default"), **Slow** and **Fast**)

If the color is set to blink, the button switches between the color defined in the **Color** field and the basic color configuration of the button (Color button)

- **Caption** text field: Use this text field to enter the caption of the button

For button styles "with frame", the text is not displayed on the button but on the frame. Tags can be included in curly brackets (e.g. {tagname})

- **Fonts** button: Click to open the "Font" dialog to specify or change the message font characteristics for each state
- **Text Blink** selection list: Click to define whether the specified text blinks and how fast it blinks for each state (**None** (no blinking, "default"), **Slow**, and **Fast**)

Unlike blinking colors, the blinking text appears and disappears

- **Write Value** selection list: Entering a value in one of the fields
If the pushbutton is in the respective state (**Open** or **Closed**), "WinStudio" writes this value to the tag specified in the **Tag/Exp** field ("Object Properties" dialog)

- **Auto Format:** If this option is enabled and if the caption includes a decimal value enclosed by curly brackets (e.g. {1.2345}) or a tag of [Real on page 113](#) type (see **Caption** above), the value is formatted according to the virtual table created by the function [SetDecimalPoints on page Set-DecimalPoints](#)

Checkbox Object

The checkbox object is an input element to activate or deactivate functions on the display.

To insert a checkbox into a WinStudio screen, click **Checkbox** in the **WinStudio Screen Tools** tool window:

1. Click in the drawing area and drag the mouse/cursor to draw the checkbox and its caption
2. Release the mouse button when the object has reached the desired size
3. Double-click on the object to open the "Object Properties" dialog.

Screens and Graphics

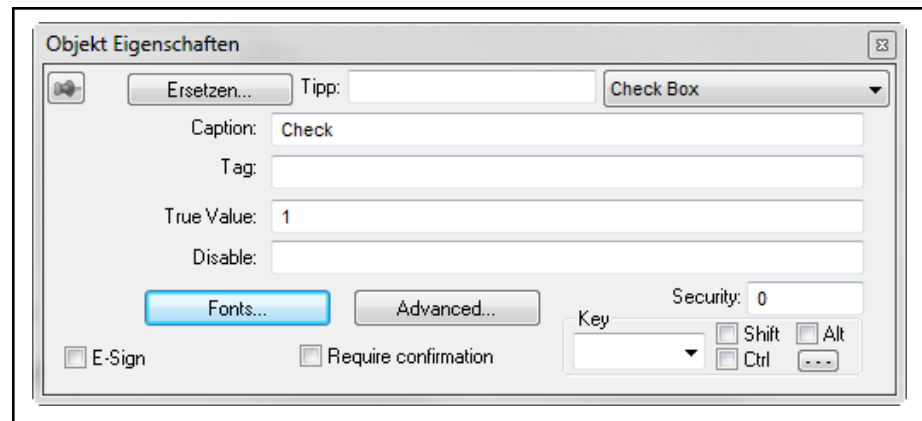


Fig. 8-30: Object Properties: Checkbox



To change the default size of the checkbox, edit the project file (**project_name.app**) to add the following setting:

[Objects]

CheckBoxSize=height_in_pixels

This setting changes the size of all checkboxes in the project and is to be used with care. This is only available for projects running in Project Viewer (Viewer.exe).

It is not available for projects running in CView or the Web Thin Client.

Use the "Object Properties" dialog to specify the following parameters for the checkbox object:

- **Caption:** Specify a caption by entering the text into the text field. Tags can be included in curly brackets (e.g. {tagname})
- **Fonts:** Specify a font style for the text by clicking the **Fonts** button
- **E-Sign:** If this option is enabled, a prompt is displayed to enter the electronic signature before executing the command
- **Require confirmation** checkbox: Click (activate) this box to ensure "WinStudio" displays a prompt to confirm the action at runtime
- **Key** selection list: Select a key from the list to connect it with the object or group of objects. This key can then be pressed at runtime to activate/deactivate the checkbox
 - Click (enable) the **Shift**, **Ctrl**, or **Alt** field to create a combination key, i.e. the <Shift>, <Ctrl> or <Alt> key has to be pressed in combination with the key specified in the selection list
 - Click the browse button ... to open the "Key Modifier" dialog to modify the combination keys. Choose **Left**, **Right** or **Left or Right** to specify the position on the [keyboard on page 73](#) of the Shift, Ctrl, or Alt combination key. If **Left or Right** is selected, the command is executed any time one of these keys is pressed in combination with the key specified in the selection list
- **Disable** field: Enter a tag or expression into this field to enable and disable the object. The checkbox object is disabled if a non-zero value is entered
- **Security** field: Enter a value into this field to specify a security level for the object, as defined under Security. If a user logs on and does not have the specified security level, "WinStudio" disables the object

Screens and Graphics

- **Tag** field: By clicking on this checkbox at runtime, the value of this tag is updated. If no **Feedback** has been specified, the value of this tag is also used to indicate the current status of the object
- **True Value** field: Specify a value that is to be used to change the control state to TRUE and to indicate that the control is in TRUE state. For further information about states, please refer to the states table
- **Advanced** button: Click this button to open the "Advanced" dialog:

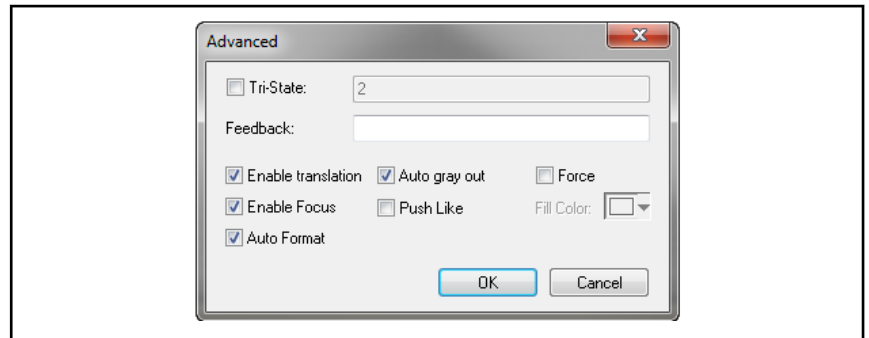


Fig. 8-31: Advanced dialog

- **Tri-State**: If enabled, the control element is assigned a third state (grayed out). The third state is displayed if the tag configured in the **Feedback** field applies the value specified in the **Tri-State** field

If the **Feedback** field is left empty, the third state is displayed if the tag configured in the **Tag** field applies the value specified in the **Tri-State** field

NOTICE

The Tri-State field cannot be configured with the same value as the True Value field. It also cannot be assigned with an empty string

- **Feedback**: Value that indicates the state of the object (FALSE, TRUE, or TRI-STATE).
If the value of the tag configured in **Feedback** is equal to the value of the tag configured in **True Value**, the state is set to TRUE
If the value of the tag configured in **Feedback** is equal to the value of the tag configured in **Tri-State**, the state is set to TRI-STATE
If none of these conditions are satisfied, the state is set to FALSE.
If the **Feedback** field is left empty, the tag configured in the **Tag** field is used as the **Feedback** tag.
- **Enable Translation**: If this option is enabled, the text is applied to the translation file.
- **Auto gray out**: Turns the caption of the object to gray if it is disabled by the **Disable** field or by the security system
- **Force**: Click (activate) this checkbox to force the tag database to recognize a tag change when clicking on the object, even if the value of the tag concerned does not change
- **Enable Focus**: If this option is enabled, the object can be activated during runtime with the navigation keys
- **Push Like**: If this option is checked the control element is displayed as a button, instead of a standard checkbox

Screens and Graphics

- **Fill Color:** Specify the fill color for the button. This option is only activated if the **Push Like** option is enabled
- **Auto Format:** If this option is enabled and if the caption includes a decimal value enclosed by curly brackets (e.g. {1.2345}) or a tag of [Real on page 113](#) type (see **Caption** above), the value is formatted according to the virtual table created by the function [SetDecimalPoints on page SetDecimalPoints](#)

Operating modes

The checkbox object can operate in two different modes: Normal and Tri-State.

For further information, refer to [Operating Modes for Checkbox and Radio Button objects on page 186](#).

Radio Button Object

The radio button object can be used to create input masks in which one of several options can be selected.

To insert a radio button into a WinStudio screen, select **Radio button** in the **WinStudio Screen Tools** tool window:

1. Click in the drawing area and drag the mouse/cursor to draw the radio button and its caption
2. Release the mouse button when the object has reached the desired size
3. Double-click on the object to open the "Object Properties" dialog.

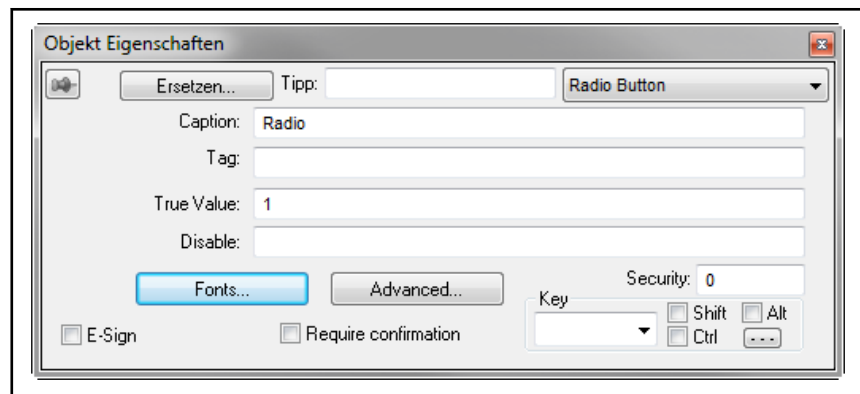


Fig.8-32: Object Properties: Radio Button



To change the default size of the radio button, edit the project file (**project_name.app**) and add the following setting:

[Objects]

RadioButtonSize=height_in_pixels

This setting changes the size of all radio buttons in the project and is to be used with care.

This is only available for projects running in Project Viewer (Viewer.exe).

It is not available for projects running in CView or the Web Thin Client.

Use the "Object Properties" dialog to specify the following parameters for the radio button object:

- **Caption:** Specify a caption by entering the text into the text field. Tags can be included in curly brackets (e.g. {tagname})
- **Fonts:** Specify a font style for the caption by clicking the Fonts button

Screens and Graphics

- **E-Sign:** If this option is enabled, a prompt is displayed to enter the electronic signature before executing the command
- **Require confirmation** checkbox: Click (activate) this box to ensure "WinStudio" displays a prompt to confirm the action at runtime
- **Key** selection list: Select a key from the list to associate it with the object or group of objects. This shortcut can then be used to activate/deactivate the radio button
 - Click (enable) the **Shift**, **Ctrl**, and/or **Alt** field to create a combination key, i.e. the <Shift>, <Ctrl> or <Alt> key has to be pressed in combination with the key specified in the selection list
 - Click the browse button ... to open the "Key Modifier" dialog to modify the combination keys. Choose **Left**, **Right** or **Left or Right** to specify the position on the [keyboard on page 73](#) of the Shift, Ctrl, or Alt combination key. If **Left or Right** is selected, the command is executed any time one of these keys is pressed in combination with the key specified in the selection list
- **Disable** field: Enter a tag or expression into this field to enable and disable the object. The radio button object is disabled by entering a value different from 0
- **Security** field: Enter a value into this field to specify a security level for the object, as defined in the security settings. If a user logs on and does not have the specified security level, "WinStudio" disables the object
- **Tag** field: By clicking on the radio button at runtime, the value of the tag entered here is updated. If no **Feedback** has been specified, the value of this tag is also used to indicate the current status of the object
- **True Value:** Specify a value that is to be used to change the control state to TRUE and to indicate that the control is in TRUE state. For further information about states, please refer to the states table
- **Advanced:** Click this button to open the Advanced dialog:

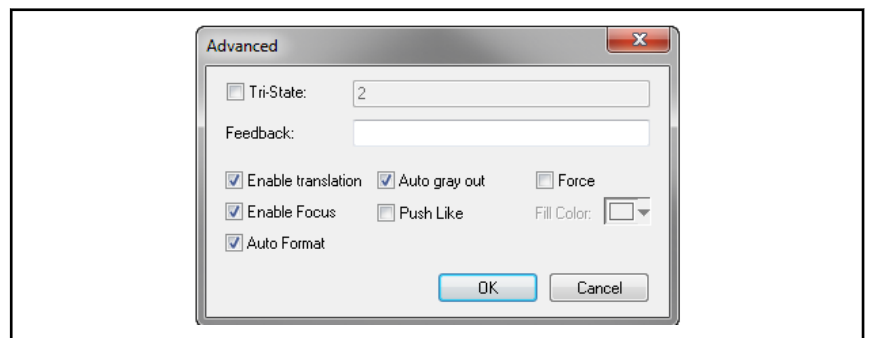


Fig. 8-33: Advanced dialog

- **Tri-State:** If enabled, the control element is assigned a third state. The third state is displayed if the tag configured in the **Feedback** field applies the value specified in the **Tri-State** field (selection field grayed out). If the **Feedback** field is left empty, the third state is displayed if the tag configured in the **Tag** field applies the value specified in the **Tri-State** field

NOTICE

The Tri-State field cannot be configured with the same value as the True Value field. It also cannot be assigned with an empty string

Screens and Graphics

- **Feedback:** Value that indicates the state of the object (FALSE, TRUE, or TRI-STATE). If the value of the tag configured in **Feedback** is equal to the value of the tag configured in **True Value**, the state is set to TRUE
If the value of the tag configured in **Feedback** is equal to the value of the tag configured in **Tri-State**, the state is set to TRI-STATE
If none of these conditions are satisfied, the state is set to FALSE.
If the **Feedback** field is left empty, the tag configured in the **Tag** field is used as the **Feedback** tag.
- **Enable Translation:** If this option is enabled, the text is applied to the translation file.
- **Auto gray out:** Turns the caption of the object to gray if it is disabled by the **Disable** field or by the security system
- **Force:** Click (activate) this checkbox to force the tag database to recognize a tag change when clicking on the object, even if the value of the tag concerned does not change
- **Enable Focus:** If this option is enabled, the object can be activated during runtime with the navigation keys
- **Push Like:** If this option is checked the control element is displayed as a button, instead of a standard radio button
- **Fill Color:** Specify the fill color for the button. This option is only activated if the **Push Like** option is enabled
- **Auto Format:** If this option is enabled and if the caption includes a decimal value enclosed by curly brackets (e.g. {1.2345}) or a tag of [Real on page 113](#) type (see **Caption** above), the value is formatted according to the virtual table created by the function [SetDecimalPoints on page SetDecimalPoints](#)

Operating modes The radio button object can operate in two different modes: Normal and Tri-State.

For further information, refer to [Operating Modes for Checkbox and Radio Button objects on page 186](#).

Operating Modes for Checkbox and Radio Button Objects

The checkbox and radio button objects both have two operating modes: "Normal" and "Tri-State".

These modes essentially work in the same way for each object.

Normal mode In normal mode, the object has two possible states: FALSE (not selected) and TRUE (selected).

Appearance of objects in normal mode

Object state	Checkbox		Radio button	
	Default	Button style	Default	Button style
FALSE	☐ Check		☐ Radio	
TRUE	☒ Check		☒ Radio	

If the object is clicked/selected at runtime, i.e. if the object state is set to TRUE, the tag configured under Object Properties is assigned the value set in the **True Value** field.

Screens and Graphics

When the object is deselected/disabled, i.e. if the object state is reset to FALSE, the value of the tag configured in the **Tag** field is NOT set to the value configured in the **True Value** field. This means in most cases that it is set to "0" or an empty string. However, there are the following exceptions:

- If the **True Value** field is set to "0", "NOT TRUE" corresponds to "1"
- If the **True Value** field is left completely empty, "NOT TRUE" corresponds to the string "UNSELECTED"

Tri-state mode Tri-state mode is enabled by selecting the **Tri-State** option in the object's "Object Properties" dialog. This adds a third possible state to the object.

Appearance of objects in Tri-State mode

Object state	Checkbox		Radio button	
	Default	Button style	Default	Button style
FALSE	☐ Check		☐ Radio	
TRUE	☒ Check		☒ Radio	
TRI-STATE	☒ Check		☒ Radio	

Clicking/selecting the object the first time works the same as it does in Normal mode.

If the object is clicked/selected a second time, the object state is set from TRUE to TRI-STATE and the value of the tag configured in the **Variable** field of the object properties is set to the value configured in the **Tri-State** field.

If the object is clicked/selected a third time, the object change depends on the type of object:

- Checkbox object is reset to FALSE from TRI-STATE

The tag in the "Variable" field is then automatically assigned the smallest positive Integer value neither configured in state TRUE nor in TRI-STATE

– Example1:

Value configured for state TRUE: "TRUE Value" =1

Value configured for the tri-state: "Tri-State" =2

Tag value in the "Variable" field for state FALSE = 0 (automatically set at runtime)

– Example2:

Value configured for state TRUE: "TRUE Value" =0

Value configured for the tri-state: "Tri-State" =1

Tag value in the "Variable" field for state FALSE = 2 (automatically set at runtime)

- The radio button object is reset from TRI-STATE back to FALSE and the value of the tag configured in the **Variable** field is set to the value configured in the **True Value** field. If it is continuously clicked on the object, the object simply toggles between TRUE and TRI-STATE

To actually clear the object, i.e. to reset the object to FALSE, click/select another radio button in the same group

This functionality is summarized in the following table:

Screens and Graphics

Current state of object	Change to the object on clicking	
	Checkbox	Radio button
FALSE	Object state set to TRUE and value set to True Value	
TRUE	Object state set to TRI-STATE and value set to Tri-State	
TRI-STATE	The object state is set to FALSE and the tag in the "Variable" field is then automatically assigned the smallest positive Integer value neither configured in state TRUE (True Value) nor in TRI-STATE	Object state set to TRUE and value set to True Value

Combo Box Object

Click on **Combo Box** in the **WinStudio Screen Tools** tool window to insert a selection list with predefined list entries.

If the number of list entries in the runtime environment exceeds the available space (list length), a scroll bar is activated for the list. If an entry is selected from the list at runtime, the combo box is minimized and the selected entry is displayed in the combo box.

Double-click on the combo box object to open the "Object Properties" dialog.

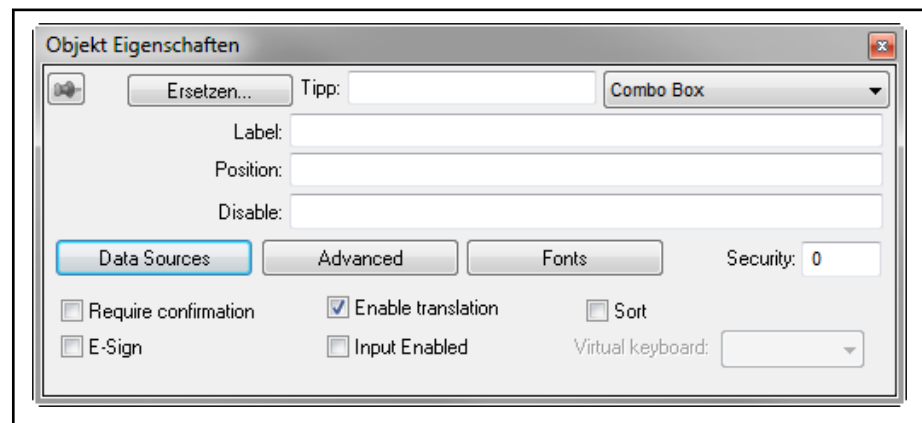


Fig. 8-34: Object Properties: Combo box

In this dialog, the following parameters can be set:

- **Label:** If a string tag is entered, the list entry currently displayed in the combo box is applied to the string tag
- **Position:** Enter a Integer-type tag in which the index of the list position currently selected in the combo box is stored. Changing this tag value changes the displayed list entry
- **Disable:** Enter a tag with a non/zero value to disable this combo box. Enter a zero or leave the field blank (default) to enable the command animation. If the combo box is disabled, it appears grayed out at runtime
- **Data Sources:** Click to open the "Data Sources" dialog (see below)
- **Advanced:** Click to open the "Combo Box - Advanced" dialog:

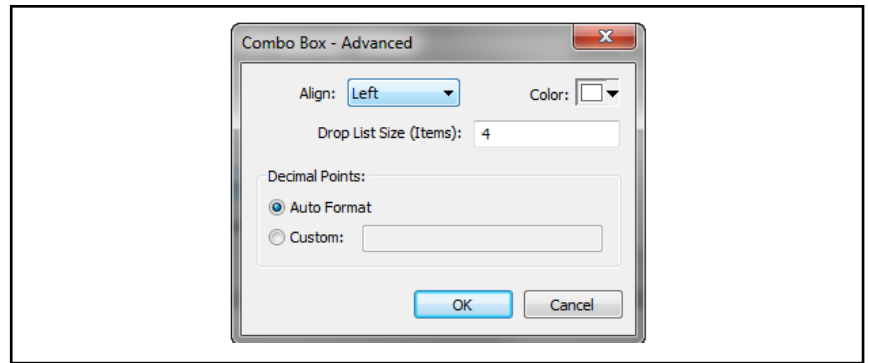


Fig. 8-35: Combo Box - Advanced

- **Align:** Select the alignment of list entries (**Left**, **Center** or **Right**). This influences the alignment in the combo box as well as in the list
- **Color:** Click to specify a background color for the combo box
- **Drop list size (items):** Enter an integer (or a tag of the [Integer on page 113](#) type) to specify the number of items that are to be displayed at one time if it is clicked on the combo box. The higher the number of items, the longer the drop down list.



If this number is less than the "Total" number of items in the list, the drop down list can be scrolled.

- "Decimal points": Select how decimal values are displayed on-screen:
 - **Auto format:** Decimal values are formatted according to the virtual table created by the [SetDecimalPoints function on page SetDecimalPoints](#)
 - **Custom:** Enter the number of decimal places to display (e.g. 2) for all decimal values
- **Fonts:** To open a fonts dialog click on [Fonts on page 244](#). Use this dialog to change the font of the list entries
- **Security:** Enter a security level for the command (0 to 255). If an operator logs on without the specified security level, the command becomes inactive. If an operator logs on with the specified security level or if this field is left empty, the command element remains active
- **Require confirmation:** Click (activate) to prompt operators to confirm a command at runtime
- **Enable translation:** Click (check) to enable automatic translation of the combo box labels using the [translation tool on page 381](#)
- **Sort:** Click (enable) to display the contents of the list in alphabetical order. This parameter is only available if the **Array Variable** type has been selected in the "Data Sources" pane
- **E-Sign:** If this option is enabled, a prompt is displayed to enter the electronic signature before activating the list selection
- **Input Enabled:** If this option is enabled, available list entries can be selected at runtime and freely selectable values can be entered
- **Virtual keyboard:** For this object, a virtual keyboard is used to enter values. Before configuration of the virtual keyboard for this object, the "Virtual Keyboard" option has to be activated in the **Viewer** tab of the project settings

Screens and Graphics

Data sources Use the "Data Sources" dialog to configure the elements/labels that are to be displayed in the combo box object.

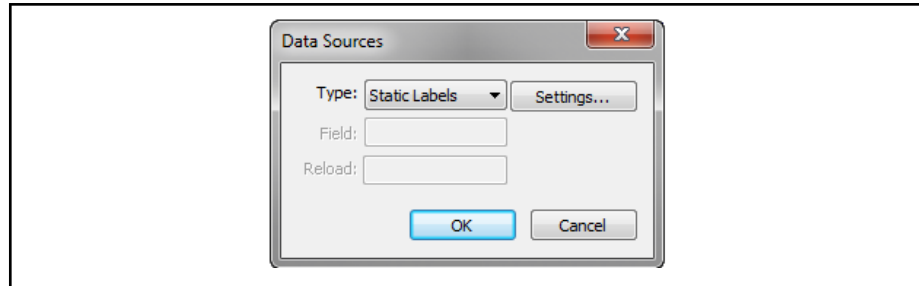


Fig. 8-36: Data Sources dialog

- **Type:** Select the type of data source to be used and click the **Settings...** button to configure the source. Each type of source is described in detail below
- **Field** (for text file and database only): Specify from which field/column the data source is to read from
- **Reload** (for text file and database only): Enter a tag name. If the value of the specified tag changes, the combo box reloads the labels from the data source

Type: Static labels If the **type** is set to **Static Labels**, the following "settings" can be configured:

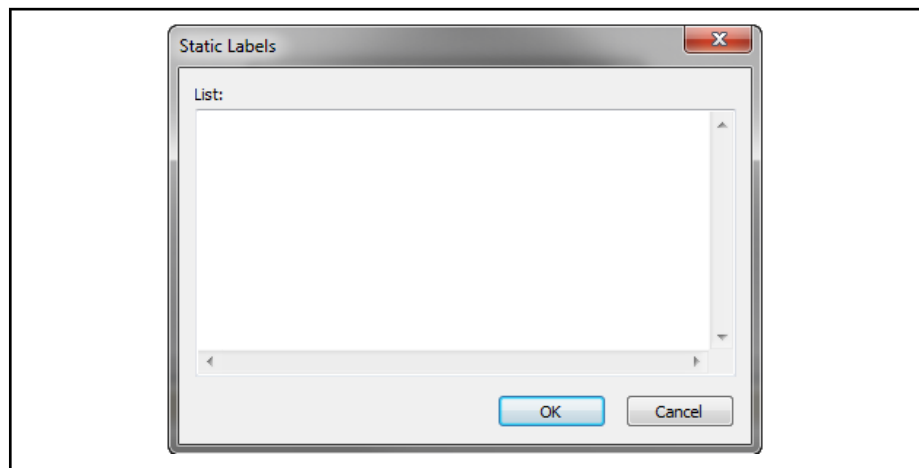


Fig. 8-37: Static Labels dialog

Enter the list entries - with one label per line - just as editing a plain text file. Tags or expressions can be used as list entries by entering them in curly brackets.

For example:

Tank#1 in {Array Nametag}.

List entries are not sorted and have to be put in the order in which they are to be displayed at runtime. The first line is position 0, the second line is position 1 and so on.

Type: Array variable If the **type** is set to **Array Variable**, the following settings can be configured:

Screens and Graphics

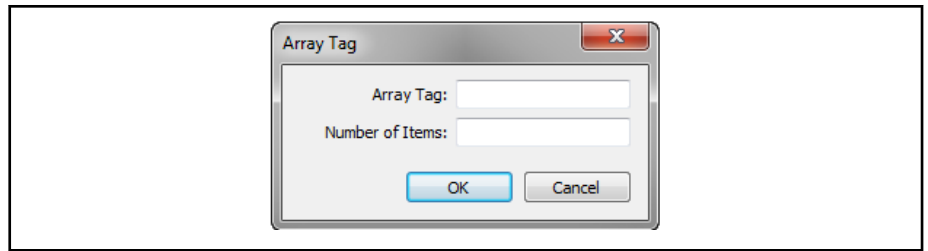


Fig. 8-38: Array Tag dialog

- **Array Variable:** Enter the name of an [Array variable on page 114](#) of the [String on page 113](#) type that contains the elements for the combo box
- **Number of Items:** Specify the number of arrays to be displayed in the combo box. "Array Index 0" is counted as the first element. If the value 4 is entered, the combo box displays the array elements 0 to 4

Type: Text file

If the **type** is set to **Text File**, the following "settings" can be configured:

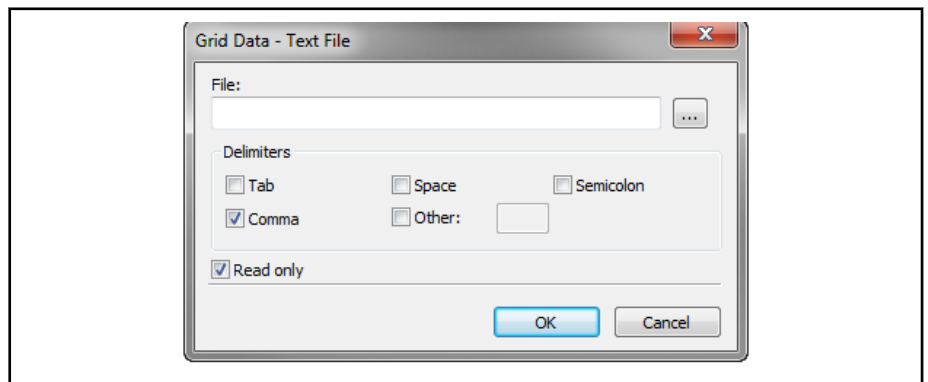


Fig. 8-39: Grid Data – Text File dialog

- **File:** Name of the source text file. Enter the file name and path or click the "..." button to browse for it. (If the file is stored in the project folder, the path in the name is not required.) Tag names can be entered in curly brackets (e.g. {Tag Name}) in the **File** field
- **Delimiters:** Set the delimiter(s) used in the data source file. For instance, if the data is to be read from a CSV (comma separated values) file, select the **Comma** option. A user-defined delimiter can be selected by checking the **Other** option and entering the user-defined delimiter in the field to the right

Type: Database

If the **type** is set to **Database**, the following "settings" can be configured:

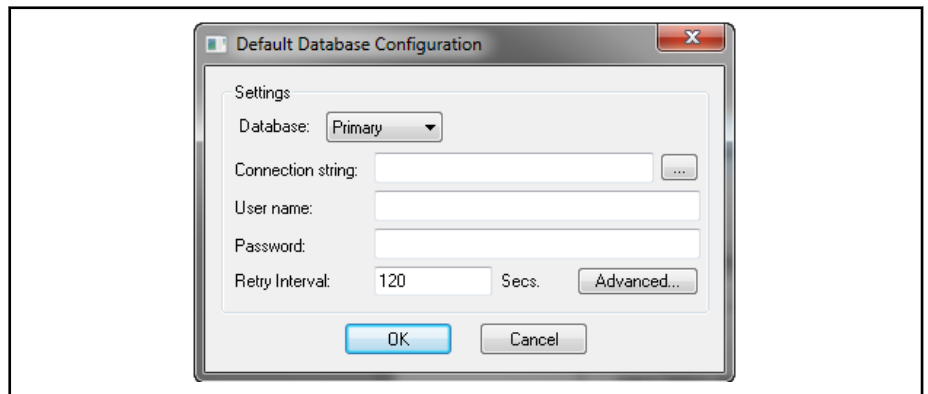


Fig. 8-40: Default Database Configuration dialog

Screens and Graphics

For further information about this dialog, refer to the [Database Configuration dialog on page 438](#).

List Box Object

To insert a list box, click on **List box** in the **WinStudio Screen Tools** tool window: Generally, if a project is executed, the "active" list box object displays a list of all messages.

On a screen containing only one list box object and no text input boxes, the list box object is activated automatically.

On a screen containing multiple list box objects and text input boxes, a cursor (pointing device) or the tab key can be used to select and activate a list box object.

A message can be selected from the active list box at runtime and the message value can be written to a tag. (If a list is too long to fit in the viewable area of a list box object, the object provides scroll bars.)

Use the option **Require enter for selection** in the "Object Properties" dialog to configure the input behavior of the list:

- Enable the **Require enter for selection** options and browse through the list of messages with the virtual keyboard, list control objects from the **Library**, pointing devices or user-defined keys with the function [Post-Keys\(\) on page 557](#). Then, use the Enter key to select the message and apply its value to the write tag. Use the ESC and Tab keys to cancel the input at any time and to return to the previously selected message
- Disable the **Require enter for selection** option to apply the value of a selected (highlighted) message automatically to the write tag

To add list box objects to a WinStudio screen, proceed as follows:

1. To insert a list box, click on **List box** in the **WinStudio Screen Tools** tool window
2. Click in the screen and drag the cursor to create/adjust a rectangle. The height of the rectangle and the font size of the displayed texts define the number of visible messages. The width of the list field defines the visible text per line. After creating the list field, the font size and style can be adjusted to display more messages in the available pane
3. Double-click on the object to open the "Object Properties" dialog.

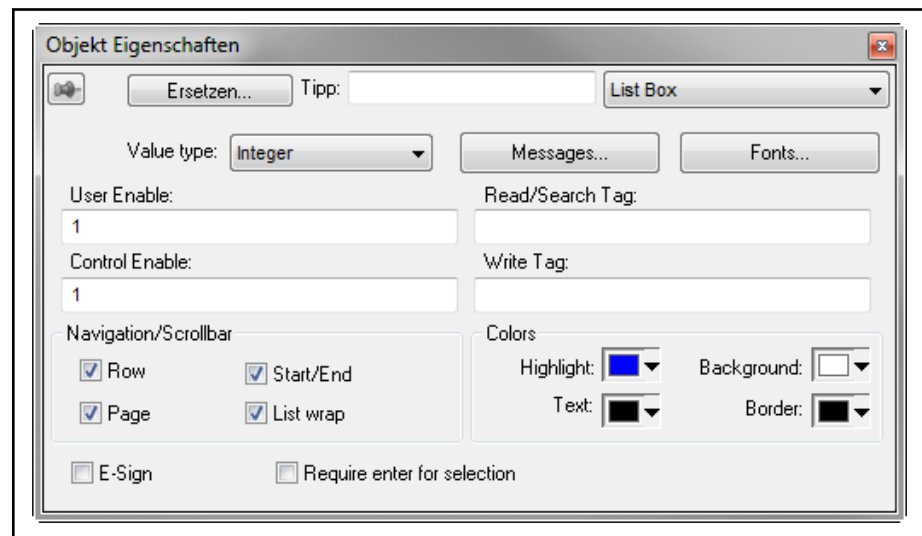


Fig. 8-41: Object Properties: List Box



The "Object Properties" dialog can also be opened, by right-clicking on the list box object or by highlighting the object, pressing the Alt+Enter keys and selecting **Properties** from the resulting context menu.

Use this dialog to specify the following parameters:

- **Value type** selection list: Select one of the following options to indicate the variable types used in the message list
 - **Boolean**
 - **Integer** (default)
 - **LSB** (Least Significant Bit)



For further information, refer to the information in the **Status** column under "[Message configuration dialog](#)" on page 194.

- **Messages:** Click to open the "[Message configuration dialog](#)" on page 194
- **User enable:** Enter a tag, expression or (non/zero) number to enable selecting messages in the runtime project by the machine operator. The default value is "1" ("true" or "enabled")
- **Control Enable:** Enter a tag, expression or ("non-zero") number to activate the "Text selection" function in the runtime environment

The default value is "1" ("true" or "enabled"). Text is selected by means of a tag in the "Read/Search Tag" field.

WinStudio uses the continuous numbers in the "Value" column under "[Message configuration dialog](#)" on page 194 to address the texts. Each number in the "Value" column represents exactly one text in the "Messages" column. If the control function is enabled, the text can be selected directly from the process without user input.

- **Read/Search Tag:** Enter an Integer or a Boolean tag to point to a selected message based on the message **value** field (under "[Message configuration dialog](#)" on page 194). Via the "Value" column, the content of the tag refers to a text in the "Messages" column.

By means of the **Control Enable** and **User Enable** fields, it can be controlled whether the operator or a process can modify this tag.

- **Write Tag** ("optional"): Enter a string tag to receive the message value of the message selected last. If the screen containing a list box object is closed and reopened, "WinStudio" uses this tag value to determine the last message selected in the list box
- **E-Sign:** If this option is enabled, a prompt is displayed to enter the electronic signature when using this object
- **Row:** Click (enable) to include row up and row down arrows in the list box object scroll bar
- **Page:** Click (enable) to include page up and page down arrows in the list box object scroll bar
- **Start/End:** Click (enable) to include start and end arrows in the list box object scroll bar
- **List wrap:** Click (enable) to enable endless scrolling through the message list

Screens and Graphics

- **Require enter for selection:** Click (enable) this option to enable selecting messages only with the Enter key. It prevents selecting messages with the Tab key
- **Color fields:** Click a color box to open the "Color" dialog or the 16-color "Color Selection" dialog. In both dialogs, colors for the list box object can be specified or changed
 - **Highlight:** Specify a color for highlighting messages (default is blue)
 - **Text:** Specify a color for highlighted text (default is black)
 - **Background:** Specify a color for the list box background (default is white)
 - **Border:** Specify a color for the list box border (default is black)

Message configuration dialog

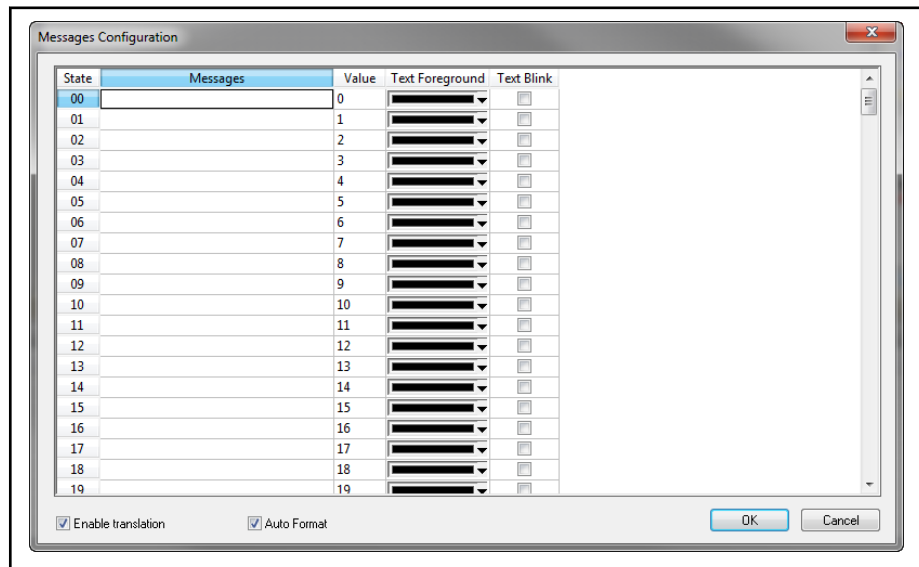


Fig. 8-42: Message configuration dialog

To use the parameters on this tab, proceed as follows:

- **Status** column ("write-protected"): Use this field to view the indexed individual messages
 "WinStudio" automatically numbers these columns depending on the settings selected in the "Value type" selection list:
 - **Boolean:** Provides two valid states, labeled 0 and 1
 - **Integer:** Provides 255 valid states, labeled 1 to 255
 - **LSB:** Provides 32 valid states (32 bits in an integer value) labeled 0 to 31
- **Messages** column: Enter the string to be displayed in the list box object. Tags can be included in messages in curly brackets (e.g. "{tagname}")
- **Value** column: Each value entered here corresponds to exactly one text in the "Messages" column. This way, it is possible to address one text via each value. If a suitable value parameterized in the "Read/Search Tag" field is written to the tag during runtime, the corresponding text is automatically selected and highlighted in the "List Object" display. Parallel to this, the value is written to the tag parameterized in the "Read/Search Tag" field. If LSB has been specified in the "Value type" field, "WinStudio" uses the values specified in the "State" column
- **Text Foreground:** Click to specify a color for the message text foreground. The color is displayed only if the message is not selected

Screens and Graphics

- **Text Blink:** Click (enable) to cause a message to blink once per second if it is selected
- **Fonts:** Click to open the "Font" dialog to change the characteristics (style, size, etc) of the message font
- **Enable translation:** Click this option to apply the text to the translation file
- **Auto Format:** If this option is enabled and if the message includes a decimal value enclosed by curly brackets (e.g. {1.2345}) or a tag of type [Real on page 113](#) (see **Message** above), the value is formatted according to the virtual table created by the function [SetDecimalPoints on page SetDecimalPoints](#)

Smart Message Object

Click on **Smart Message** in the **WinStudio Screen Tools** tool window to create one or more smart message objects to display messages and graphics when executing the project. "WinStudio" provides the following "Smart Message object types":

- **Message display:** Enables displaying any one of multiple messages within a single message object
- **Multistate indicator:** Enables display of one of multiple messages within a single message object while additional bitmap graphics can be displayed in combination with the messages
- **Multistate switch:** Enables displaying messages and bitmap graphics. This object also resembles a multi-position switch as it enables incremental switching through messages by clicking on the object at runtime

These "Smart Message object types" are distinguished due to their different scope of functions in configuration. However, they have one parameter in common: The variable in the "Read Tag/Expression" field for message selection.

To add a "Smart Message Object" to a WinStudio screen, proceed as follows:

1. Click on **Smart Message** and position the cursor in the screen editor
2. Create a rectangle by clicking and drawing. The size of the rectangle and the font size define how much text can be displayed as well as the size of the bitmap graphic. The rectangle size and font characteristics can be changed later on to allow longer messages to be displayed
3. Double-click on the object to open the "Object Properties" dialog.

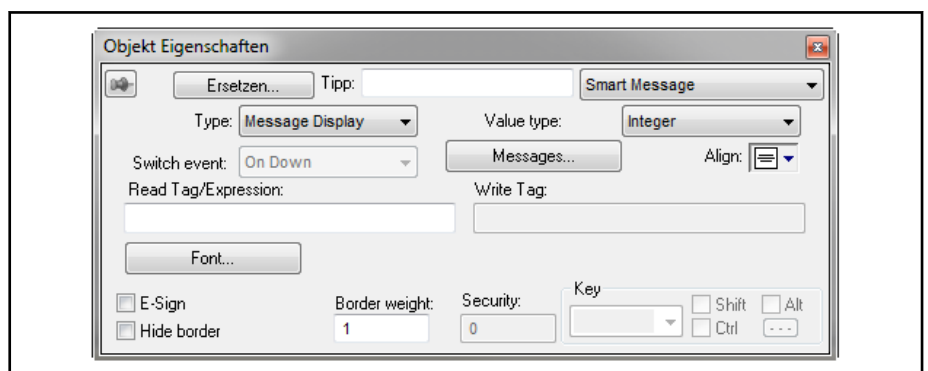


Fig. 8-43: Object Properties: Smart Message

Use this dialog to specify the following parameters:

- **Type:** Click to select the object type. The object type sets the behavior of the object at runtime and the features supported by it:

Screens and Graphics

- **Message display** (default)
- **Multistate indicator**
- **Multistate switch**
- **Value type** selection list: Select the type of values used to index the message list:
 - **Boolean**: Provides two valid states. Use this selection to display one of two different messages, based on a Boolean value (0 or 1)
 - **Integer** (default): Provides 500 valid states. Use this selection to display different messages based on numeric values from an Integer tag
 - **LSB** (least significant bit): Provides 32 valid states (32 bits in an Integer value). Use this selection to display different messages based on which bit from an Integer tag is set. If more than one bit from the Integer tag is set simultaneously, the message associated with the lowest value bit is displayed



If **Multistate switch** is selected as **Smart message** object, only 16 different messages can be associated with the object, even for **Integer** or **LSB** values.

- **Read tag/expression**: Enter a tag or expression. The value determines which message is displayed by the object at runtime
- **Write tag** (optional and available for **Multistate switch** only): Enter the name of a project tag. The value associated with the message currently displayed is written to this tag
- **Align**: Select the alignment of the text displayed by the "Smart message object"
- **Write tag** pane (keyboard shortcuts available for **Multistate switch** only): Keyboard shortcuts to select the next message on the keyboard if the "Multistate switch" type has been selected. This option is particularly useful if the keyboard is the only available physical interface to interact with the user interface at runtime
- **Switch event** selection list (available for **Multistate switch** only): Select one of the following options to specify when the message is changed:
 - **On down**: Switch to the next message when clicking on the object (default)
 - **On while**: Switch to the next message continuously while holding the mouse button down on the object
 - **On up**: Switch to the next message if the mouse button on the object is released
- **E-Sign** (available for **Multistate switch** only): If this option is enabled, a prompt is displayed to enter the electronic signature before executing the action
- **Security** (available for **Multistate switch** only): User right required for object/animation
- **Hide border**: If this option is enabled, the line border of the object is not visible
- **Border weight**: Defines the thickness of the line drawn around the object (the border)
- **Fonts**: Opens the Fonts dialog to configure the font settings for the text displayed in the object

Screens and Graphics

- **Messages...:** Opens the "Configuration" dialog to configure the messages for the object, as follows:

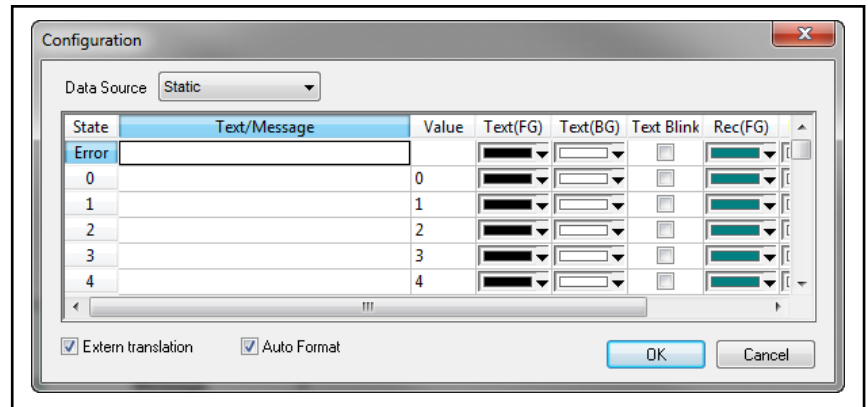


Fig.8-44: Smart message configuration dialog

- **Data source:** The messages displayed by the object can either be entered directly into the table (**Static**) or read from an external text file (**Text file**). If the **Static** data source type is selected, the "Configuration" dialog is displayed as illustrated above and all settings can be configured in the table.

If **Text file** is selected, the "Configuration" dialog displays a field to enter path and name of the file from which the messages are to be read (source file).

For further information about text file formatting, refer to ["Source file format" on page 198](#).

- **Enable translation:** If this option is enabled, the text is applied to the translation file.
- **Auto format:** If this option is enabled and if the message includes a decimal value enclosed by curly brackets (e.g. {1.2345}) or a tag of type [Real on page 113](#) (see **Text/Message** below), the value is formatted according to the virtual table created by the function [SetDecimalPoints on page SetDecimalPoints](#)

The following table describes the meaning of the properties associated with each message, regardless of the data source:

Property	Description
Text/Message	Message (text) that is displayed when selected during runtime. Tags can be included in the message in curly brackets (e.g. {tag name})
Value	Each message has to be assigned a unique value. If a suitable value is written to the tag during runtime, which is parameterized in the "Read Tag/Expression" field, the corresponding text is automatically selected and displayed. If there is no suitable message, the message configured in the first line (State = Error) is displayed during runtime. If Multistate switch has been specified as object type , the value of the current message is also written to a tag if configured in the "Write tag" field
Text (FG)	Foreground color for the messages displayed at runtime
Text (BG)	Background color for the messages displayed at runtime
Text blink	If enabled, the message text blinks at runtime
Rec (FG)	Line color (border) for the rectangle behind the message
Rec (BG)	Background (fill) color for the rectangle behind the message
Rec blink	If enabled, the rectangle behind the message blinks during runtime

Screens and Graphics

Property	Description
Graphic file	Path and name of the bitmap file (*.BMP) (if any) that is displayed when the assigned message is selected during runtime. If the path is not specified, the bitmap file has to be stored in the project folder
Transparent	Select the color to be transparent in the graphics file if the En. transparent checkbox is enabled
Transparent enabled	If enabled, the color selected in the Transparent field is set as transparent in the graphics file



The properties **Graphics file**, **Transparent** and **Transparent Enabled** are not available for the **Message Display** type.



Data can be copied from this dialog and pasted into an Excel worksheet and vice versa.

Source file format

This section describes the format of the text file supported by the "SmartMessage" object if a text file is configured as data source. The main advantage of using an external text file instead of static values is that it provides the flexibility to change the messages during runtime by changing the name of the used text file or even the content of the text file.

The text file has to be created in CSV format (Comma Separated Values), where the comma character (",") is used to divide the columns (data) in each line of the file. Therefore, any CSV editor such as Microsoft Notepad and Microsoft Excel can be used to create the CSV file with the messages and their properties for the "Smart Message object".

For a description of all message properties, refer to [Smart Message object on page 195](#). The order of the data in the CSV file is described in the following table:

Column #	Property	Default value
1	Text/Message	-
2	Value	-
3	Text (FG)	0
4	Text (BG)	16777215
5	Text Blink	0
6	Rec (FG)	8421376
7	Rec (BG)	16777215
8	Rec Blink	0
9	Graphic File	-
10	Transparent	0
11	Transparent enabled	0

When configuring text messages with a comma character as part of the message, the entire message has to be configured between quotation marks (e.g. "Caution, switch off the motor"). Otherwise, the comma is interpreted as data separator and not as part of the message.

- The first line of this file is equivalent to the **State = Error**. In other words: If the configured tag in the "Read tag/expression" field is given a value

to which no text has been assigned, the message configured in the first line (**State = Error**) is displayed.

As the error text of the first line is not triggered by a specific value, the entry in the "Value" column is irrelevant. This first line always has to be configured, regardless of the object type (even for **Multistate switch**).

- The **Text/Message** and **Value** columns are mandatory. All other columns are optional. If no values are specified here, the default values are used (refer to the table).
- The fields **Text(FG)**, **Text(BG)**, **Rec(FG)**, **Rec(BG)** and **Transparent** can be configured with the code of the color associated with it. The code can be entered directly in decimal format (e.g. 255) or in hexadecimal format using the syntax #value (e.g. #0000FF).
- The fields **Text Blink**, **Rec Blink** and **En. Transparent** can be configured with Boolean values "0" or "1" (0 = **Disabled**; 1 = **Enabled**), or with the keywords **FALSE** or **TRUE** (**FALSE** = **Disabled**; **TRUE** = **Enabled**).

Example:

```
Error Message, ,0,16777215,1,8421376,16777215,1,error.bmp,0,0
Message Zero,0,0,16777215,0,8421376,16777215,0,open.bmp,65280,1
Message Ten,10,0,16777215,0,8421376,16777215,0,closed.bmp,65280,1
Message Twenty,20,0,16777215,0,8421376,16777215,0,,0,0
Message Thirty,30,0,16777215,0,8421376,16777215,0,,0,0
```



The editor of the "SmartMessage" object - when selecting
Data source = Static

- can be used to configure messages, values and colors. Optionally, the content of the table can be highlighted and inserted into an Excel worksheet via the Windows clipboard (Copy&Paste).

Afterwards, the Excel worksheet can be saved as CSV file (**File ► Save As**) and implemented again in WinStudio by switching the data source to "Text File".

This procedure provides a user friendly interface for configuring the color codes.

8.3.4 WinStudio Libraries

Symbols

Symbols - General Information

The **library** is a visual browser for all of the [symbols on page 200](#) that are available to be inserted in a WinStudio screen.

To open the "WinStudio Symbols" tool window, select the entry **WinStudio library** in the context menu **WinStudio ► Tools** or in the context menu **User Symbols**

Screens and Graphics

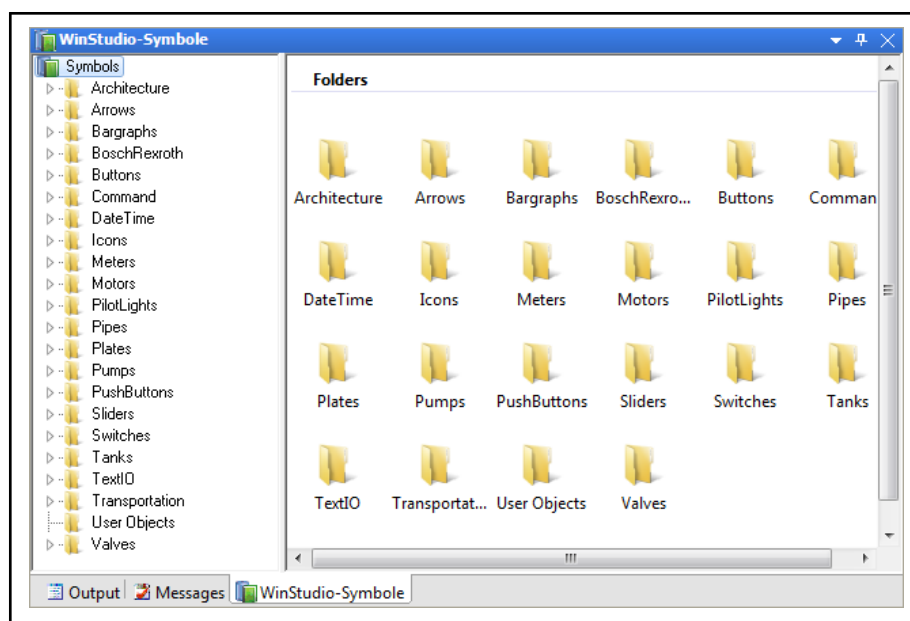


Fig.8-45: Symbols Library window

The "WinStudio Symbols" tool window offers a comprehensive selection of predefined symbols. These symbols are sorted in folders according to categories (e.g. buttons, counters, tanks).

To select a symbol and insert it into a WinStudio screen, proceed as follows:

1. Find the required symbol in the library and click on it
The mouse cursor changes to indicate that a symbol has been selected
2. Return to the WinStudio screen in which the symbol is to be inserted
3. Click anywhere in the screen to position the selected symbol
4. If required, configure the [object properties on page 164](#) of the symbol

For further information, refer to [Saving Own Project Symbols on page 200](#).

Providing user-defined symbols to other projects

User-defined symbols are only available in the project where they were initially created and saved, i.e. the "User Symbols" folder in the project tree is project-specific. However, user-defined symbols can be sent to the "User Objects" folder of the WinStudio symbol library to make them available for all projects.

1. Select the symbol in the "User Symbols" folder that is to be made available for other projects
Select the entry **Apply to WinStudio library** in the context menu of this element
2. By means of this function, the symbol is copied to the "..\Users\All Users\WinStudio\Symbols" directory and now appears in the "User Objects" folder

Saving Own Project Symbols

A **Symbol** is a group of one or multiple objects saved in the "User Symbols" folder that can be reused without limits in the project.

Every time a symbol is used, a copy is made that is linked to the master **Symbol** file in the "Symbols" folder.

These linked copies are also referred to as "instances" of the Symbol.

If any changes are made to the master symbol afterwards, these changes automatically apply to every linked copy in all WinStudio screens.

Screens and Graphics

Each linked copy of the master symbol can be individually extended by **user-defined properties**.

For example, if a gauge to display tank filling levels is created and saved as master symbol, each instance can be supplemented individually:

Each instance of the gauge can display the filling level for a different tank.

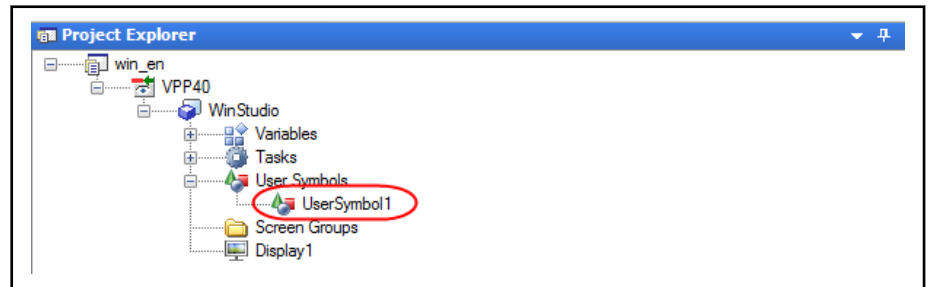


Fig.8-46: "User Symbols" folder

Creating a master symbol

To create a master symbol and save it to the "User symbols" folder, proceed as follows:

1. Design the symbol just as normally editing a WinStudio screen using any combination of [static on page 167](#) and [active objects on page 174](#).
For example, three [checkboxes on page 181](#) in a [rectangular pane on page 170](#):

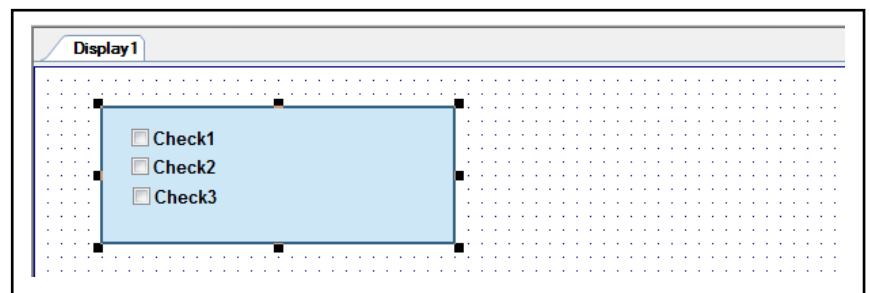


Fig.8-47: Adding objects into a WinStudio screen

2. Select the object(s) or [group on page 235](#) to be saved as a symbols.

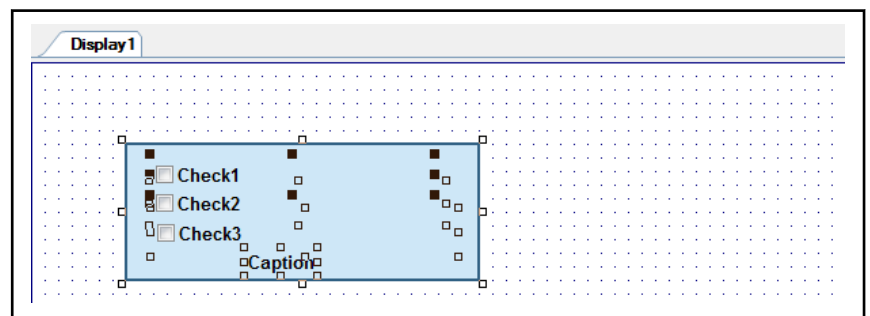


Fig.8-48: Selecting the objects

It is not necessary to make a group out of two or more objects before saving them as a symbol. Saving the objects together as a symbol effectively groups them.

However, there is one use case requiring grouping the objects:

A symbol can only display a single tooltip.

If a symbol is generated from several objects without previous grouping, existing (if any) tooltips cannot be displayed (under [object properties on page 164](#)).

Screens and Graphics

To specify a tooltip for the symbol, the objects have to be grouped first to configure the tooltip for the group. This tooltip is retained if the group is saved as a symbol.

- Right-click on the selection and click on **Create Linked Symbol** in the context menu.

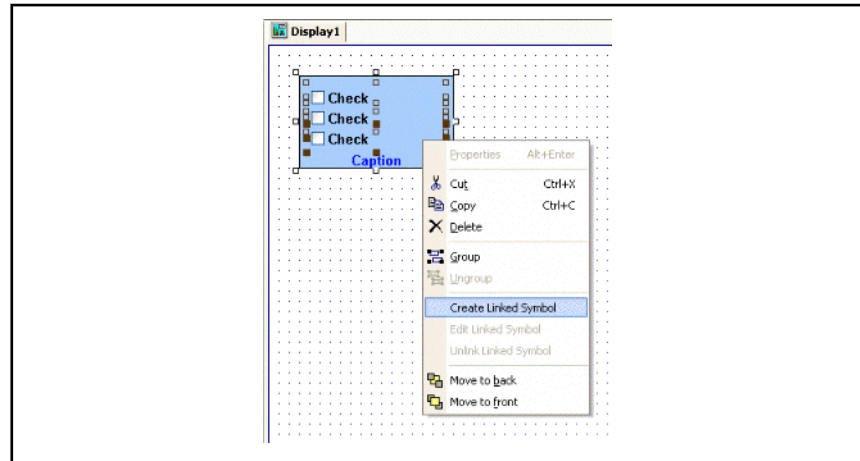


Fig.8-49: Creating a linked symbol

- A standard "Save As" dialog is displayed to specify a file name for the new symbol. Symbol files (".sym") are saved in the **\Symbol** folder of the project.

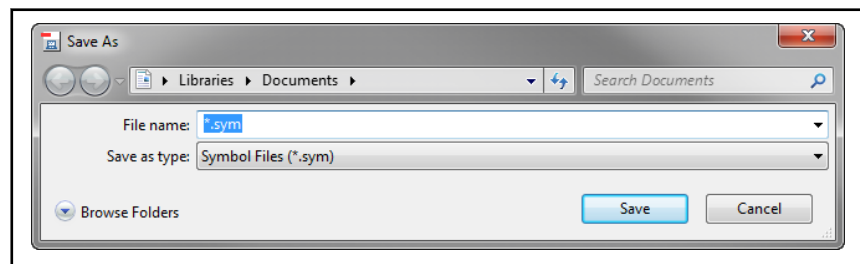


Fig.8-50: Saving the symbol file test.sym

- Click **Save** to save the file.

The symbol appears in the "User Symbols" folder, in the [Graphics tab on page 156](#) of the Project Explorer.

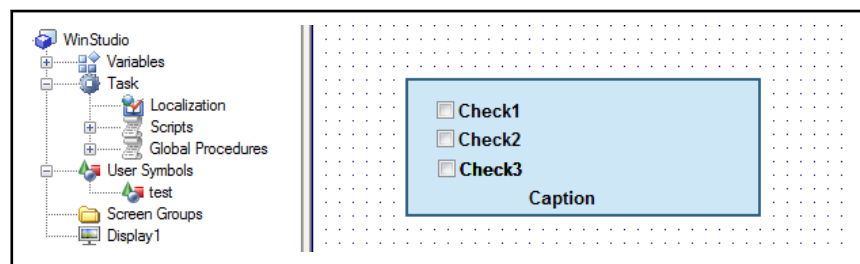


Fig.8-51: Symbol file in the Project Explorer

The symbol is now ready to be reused in the project. However, the way it is currently saved, every copy has identical properties.

Now, user-defined properties have to be defined for the symbol, i.e. the properties that can be adjusted every time the symbol is used.

Editing the master symbol

Master symbols can be edited after saving it, e.g. by adding or deleting objects in the symbol or by defining user-defined properties.

Screens and Graphics

Remember that any changes to the master symbol are automatically applied to any linked copies (instances) in the entire project. For example, the font color was changed from black to blue...

There is one exception:

A Change of the tooltip for a symbol as described in "[Creating a master symbol](#)" on page 201, only applies to new instances of this symbol. Existing instances are not changed.

To edit a symbol, proceed as follows:

1. In the "User Symbols" folder, double-click on the symbol file.



Optionally, right-click on any instance of the symbol and choose **Edit Linked Symbol** from the context menu.

The symbol file is opened for editing in a separate window.

This symbol editor works in the same way as the WinStudio screen editor, except that every object in the window is part of a symbol.

If objects are added, moved or deleted in the symbol editor, the size or shape of the symbol may be changed and influence the layout of the WinStudio screens it is used in.

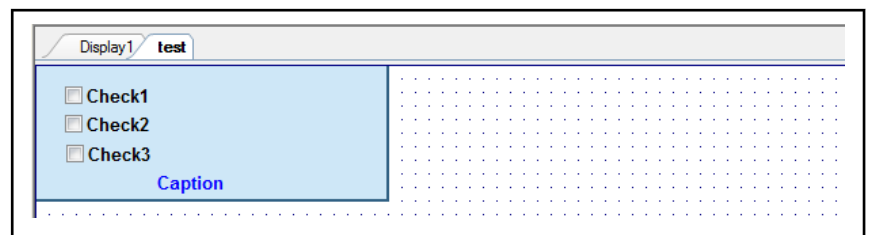


Fig. 8-52: Symbol file opened for editing

Besides adding, moving or deleting objects in the symbol, the [object properties on page 164](#) can be edited just as usual.

It is not always useful that all properties of a symbol are similar in all instances.

In this example, the stored [tags on page 106](#) and the labeling of the three checkboxes are to be freely selectable for each instance.

2. Select the first object of the symbol and open its object properties. For example, the first checkbox:

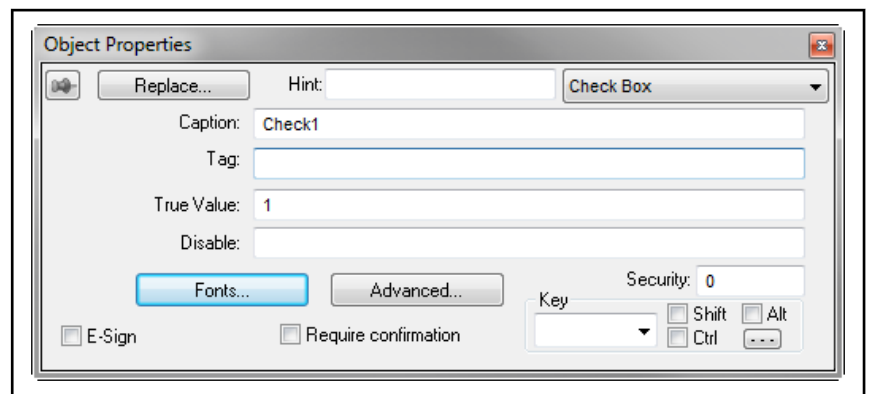


Fig. 8-53: Object Properties for the first check box

3. In any field where usually a tag, expression, or value is to be configured, a user-defined property can be defined using the syntax:

Screens and Graphics

[Category.] Property: [Value]

...where:

- **Category** is an "optional" name for a collection of properties to be grouped, such as all captions or checkbox values. If no **Category** is specified for a property, it is automatically listed under the "Main" category
- **Property** is the display name to describe a property. A display name is required for each property, and has to be always followed by a colon (":")
- **Value** is an "optional" default value for the property



All tag/expression syntax rules apply to "value", including tag names on page 111, pointers on page 119, arrays on page 114, strings, numerical and Boolean values and scripting functions.

In the following example, the tag containing the current state of the checkbox is to be individually set for every master symbol instance.

In the **Tag** field, enter **#Option1.Tag**; as illustrated.

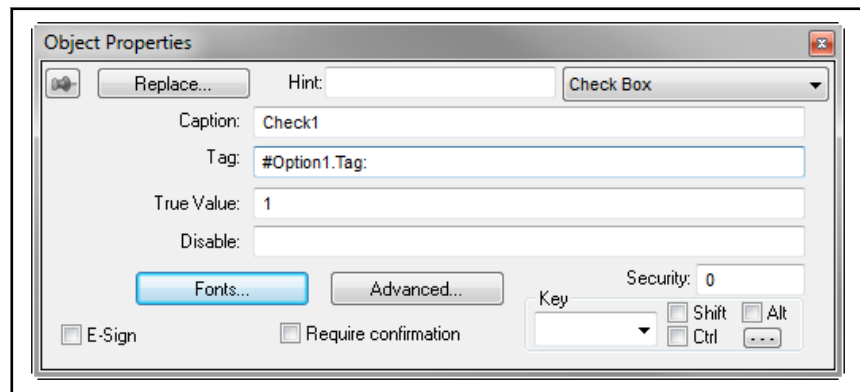


Fig.8-54: Defining a user-defined property for the tag field

The entry "#Option1.Tag:" in the master symbol corresponds to the following display in the Object Properties dialog of a symbol instance:

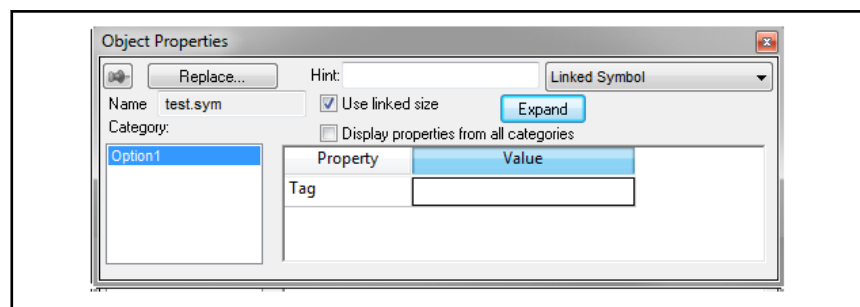


Fig.8-55: User-defined properties of a symbol

A tag has to be entered in the "Value" column to obtain the status of the checkbox for this instance. For further information, see below.

- Object property fields usually expecting simple texts (strings) can be replaced by dynamic elements (placeholders) in curly brackets. In this example, the **Caption** field requires a string. As illustrated, enter **{#Option1.Caption:"Option 1"}**.

Screens and Graphics

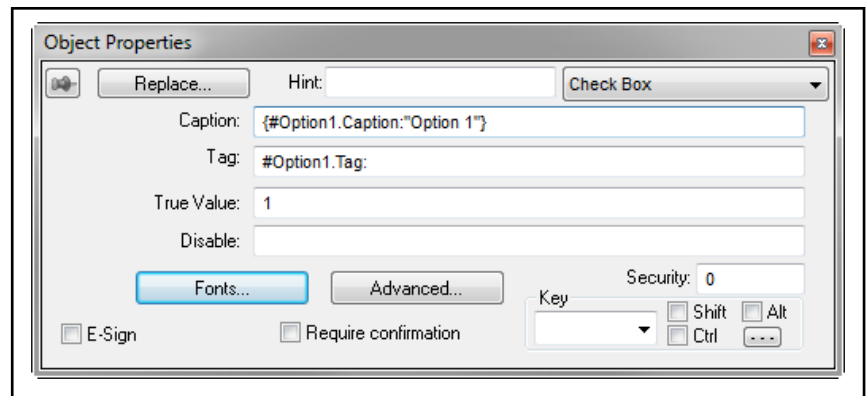


Fig.8-56: Defining a user-defined property for the Caption field

The entry in the master symbol leads to the following entry in the Object Properties dialog of the symbol instance:

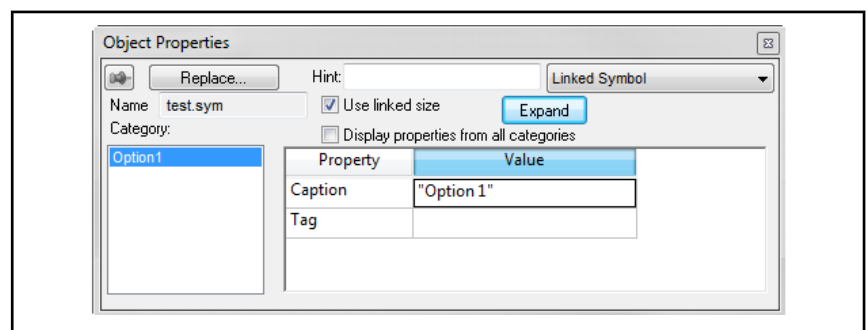


Fig.8-57: User-defined properties of a symbol

5. Repeat steps 2 to 4 to parameterize the remaining user-defined properties of the master symbol. In this example, the Object Properties dialog of the symbol instance displays the following entries:

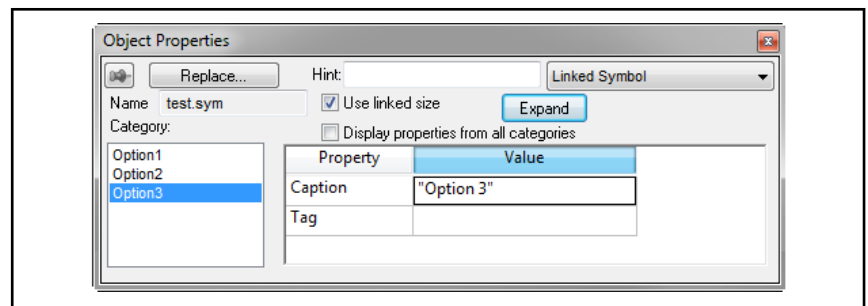


Fig.8-58: User-defined properties of a symbol



Display properties from all categories enables displaying all object properties of the symbol in a list.

6. Save the symbol and close the symbol editor.
7. Afterwards, execute "Verify Application" in the context menu of the "WinStudio" folder. Thus, all existing instances of the symbol in the project are updated.

Adding tooltips to user-defined properties

For each user-defined property available in the master symbol, a description can be defined simplifying the configuration of the symbol instances.

Screens and Graphics

After creating a symbol, open it with the symbol editor, right-click in the symbol editor (not on the symbol itself) and choose **Edit Symbol Properties** from the context menu.

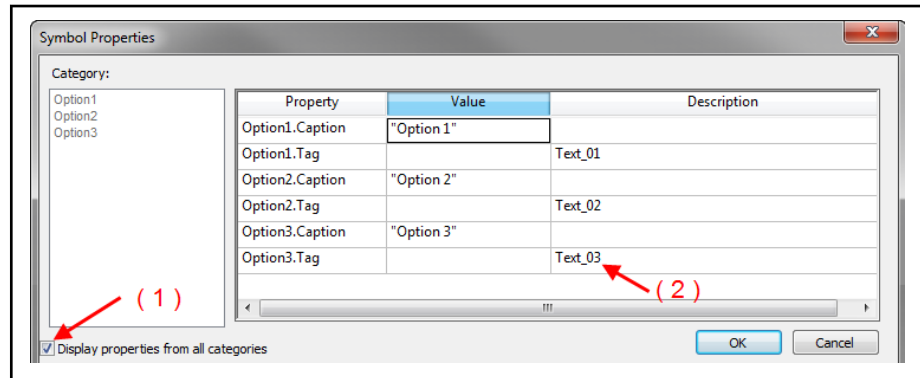


Fig. 8-59: Edit symbol properties

By means of checkbox (1), **Display properties from all categories**, information of all objects can be displayed in a list.

In column (2), **Description**, a text is to be entered that is later used as tooltip.

When inserting and configuring a symbol instance in the screen, the description is displayed as tooltip if the mouse cursor is moved over the name of the property as displayed in the following figure:

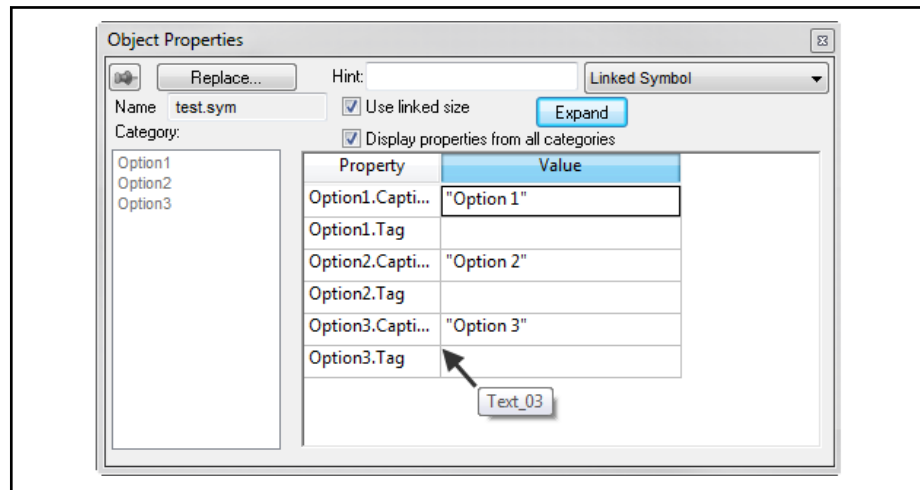


Fig. 8-60: Tooltip indicating the description of the property

Symbol password protection

Passwords can be defined for all user-defined symbols to prevent unauthorized editing or analysis. To protect a symbol, proceed as follows:

1. In the "User Symbols" folder, right-click on the desired symbol file (.sym) and choose **Change Password** from the context menu. The "Edit Protection" dialog is displayed
2. Enter the password in the **New Password** field
3. Enter the password again into the **Confirm Password** field
4. Click on **OK**

Once this is done, a prompt for the password is displayed whenever the symbol is to be edited or the link to an instance of the symbol is to be removed.

Adding symbols into a WinStudio screen

To insert a symbol into a screen and complete its user-defined properties, proceed as follows:

Screens and Graphics

1. Open the desired screen from the [Screens folder on page 156](#) or insert a new screen
2. To open the symbols library, proceed as follows:
 - Open the context menu **WinStudio** ► **Tools** and select "WinStudio library" or
 - Open the context menu **User Symbols** and select "WinStudio library" or
 - Right-click on the WinStudio screen editor where the symbol is to be inserted and click **Insert Linked Symbol** in the context menu
3. Select the symbol from the symbols library and click on the WinStudio screen:

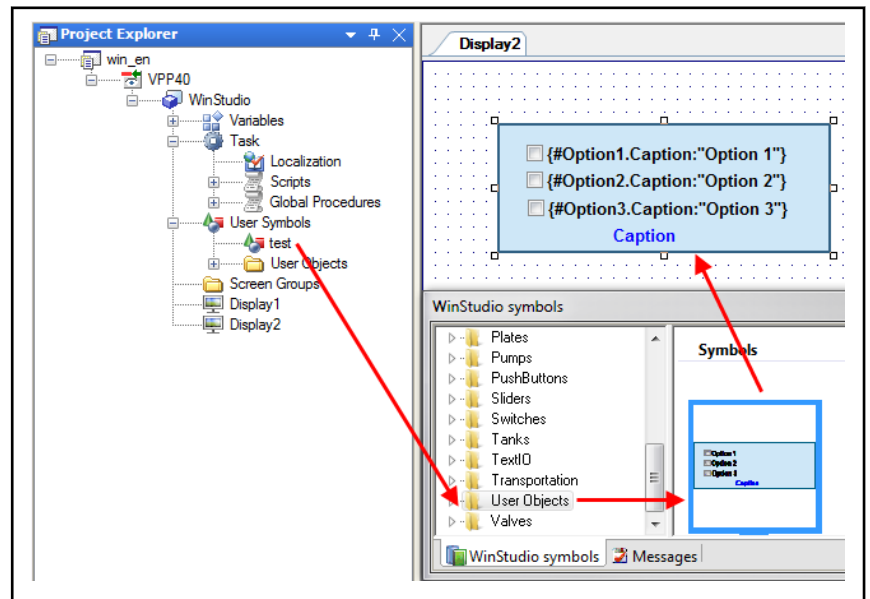


Fig.8-61: Positioning a symbol in a SCREEN

If the symbol is inserted it can be edited like any other object in the screen. It can be [aligned and distributed on page 233](#) with other objects and [animations on page 220](#) can be applied

However, first of all, the user-defined properties for this instance of the symbol have to be completed

4. Open the Object Properties for the symbol
5. Assign real process values (texts in quotation marks or tag names) to the "Tag" transfer parameters

Screens and Graphics

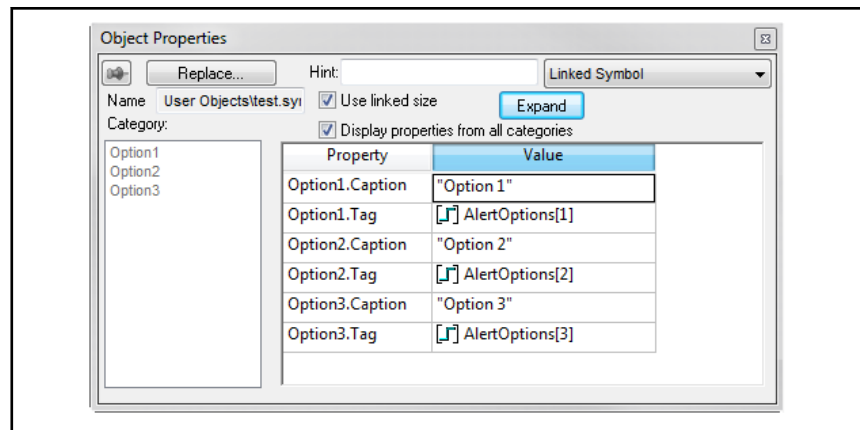


Fig.8-62: Completed properties for the symbol

- Click **OK** to close the "Symbol Properties" dialog and close afterwards the "Object Properties" dialog

At runtime, the placeholders in Engineering {} are replaced by the true (assigned) process values (texts, tag values).

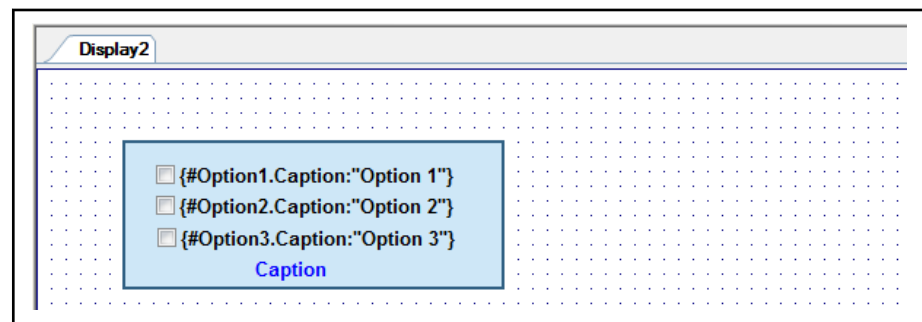


Fig.8-63: Symbol during editing

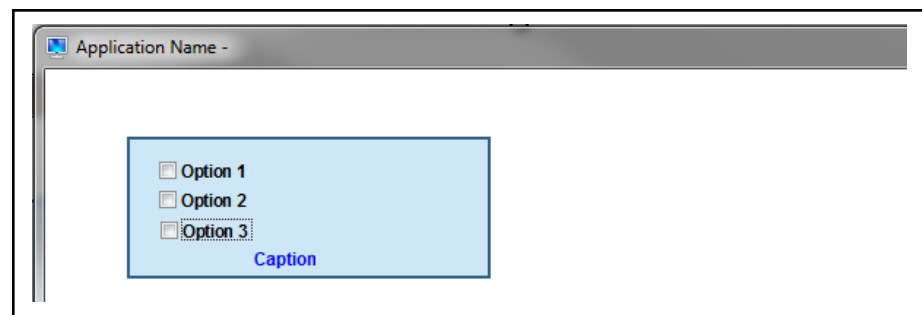


Fig.8-64: Symbol at runtime



Remember that the assigned process values for each instance of a master symbol are independent of all other instances of this master symbol.

However, if changes are made to the master symbol file, these changes are automatically applied to all instances.

ActiveX Control

In the **WinStudio Screen Tools** tool window, click on **ActiveX control**. The "Insert ActiveX control" dialog is displayed to select ActiveX elements and insert them into WinStudio screens.

If the dialog is displayed, it provides a list of all ActiveX components that are registered on the PC.

Screens and Graphics

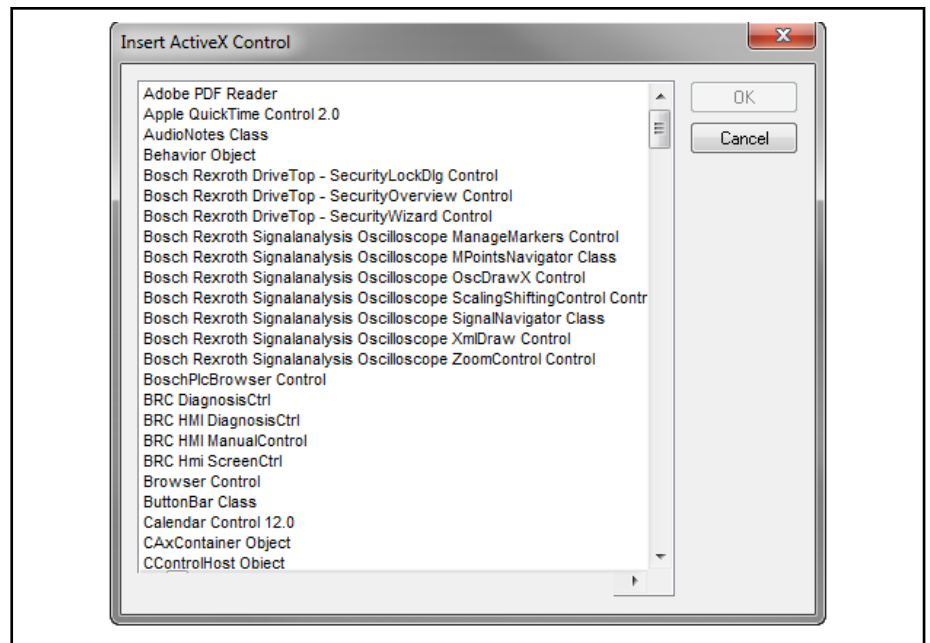


Fig. 8-65: Insert ActiveX control dialog



If the ActiveX controls are to be used in the project, the runtime stations are to have the same controls already installed and registered.

As the "Automatic Download" and "Automatic Registration" functions are mostly deactivated on target devices for safety reasons, ActiveX controls are not always available after project download.

To be able to use ActiveX controls despite these security settings, they have to be transferred to the target device and registered manually. If the ActiveX control is registered for the first time, it can be updated with the project by adjusting the project file (projekt_name.app):

If the automatic download of ActiveX control elements is still to be enabled, this can be achieved by manually editing the project file (projekt_name.app) to add the following settings:

Project file "projekt_name.app"

```
[UsedControls]
EnableDownload=1
Count=number of controls

[Used Control1]
CLSID= class ID of the ActiveX control
Version = version of the ActiveX control
Codebase= URL of the ActiveX control file, or of the .CAB file that contains the ActiveX control files
RegFile1= name of a specific .OCX or .DLL file within the .CAB file; see below
RegFilen= name of a specific .OCX or .DLL file within the .CAB file; see below
...
[UsedControln]
...
```

The "CLSID" and "Version" settings are required for each ActiveX control and have to match the ID and version of the actual control file(s) to which "Codebase" is linked.

This allows a runtime station to check the ActiveX control against elements that are already registered. If the settings do not match, the runtime station may unnecessarily download the same control element again.

Screens and Graphics

If the "CLSID" and "Version" settings for an ActiveX control are unknown, they can be found in the registry key of an already installed and registered control element. Search for the control file in **HKEY_CLASSES_ROOT\CLSID** in the Windows Registry.

The URL for the "Codebase" setting can be either absolute or relative to the "home" directory of the web server.

For example:

```
Codebase=http:// "server_adresse" /AddOns/IndDateTimePick.ocx
...OR...
Codebase=AddOns/IndDateTimePick.ocx
```

The "Regfile" settings are only required if "Codebase" is linked to a .CAB file (container for multiple ActiveX controls). In this case, a "Regfile" entry has to be created for each received ActiveX control.

ActiveX controls are components designed according to a standard. These elements can be implemented into "Rexroth WinStudio" application screens and executed at runtime. ActiveX controls can provide the following interfaces:

- **Properties:** Variables whose values can be read and/or written for the project (e.g. object color, file name, URL, etc.)
- **Methods:** Functions from the ActiveX object that can be triggered by the project (e.g. open dialog, execute calculation, etc.)
- **Events:** Internal messages that can trigger the execution of expressions in the project (e.g. Mouse_Click, Download_Completed, and so forth)

The names of the properties, methods and events supported by each ActiveX object depend on its own implementation.

There are two different ways to interface the project with the ActiveX control elements:

- By using the ActiveX function
[XGet\(\) on page 684](#),
[XSet\(\) on page 685](#) and
[XRun\(\) on page 684](#)
 or
- by using the "Object Properties" window to configure the object

Double-click on the ActiveX control to open the "Object Properties" dialog.

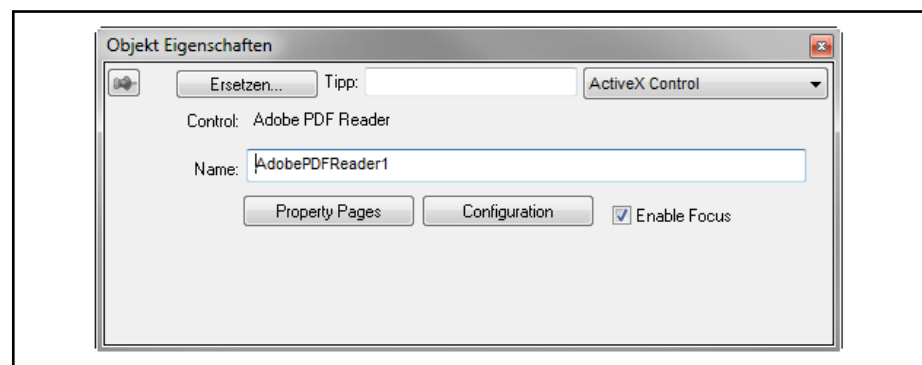


Fig.8-66: Object Properties: ActiveX Control

The "Object Properties" window displays the name of the ActiveX control. Generally, each ActiveX control is either a *.dll or a *.ocx file registered on the local computer.

Screens and Graphics

Assign the ActiveX control a unique name (alias) in the "Name" field (e.g. My-Control). This name is required for function calls from the scripting language.



ActiveX controls can never be configured with the same name in one screen.

The **Property Pages** button opens the standard window for configuring the static properties (if available). The layout and the options in this dialog depend on the implementation of each ActiveX control. Use this interface to set properties that are not to be changed during runtime (fixed properties).

The **Configuration** button on the "Object Properties" window opens a configuration dialog that enables the following tasks:

- Connect tags to properties of the ActiveX control
- Trigger methods from the ActiveX control based on tag change
- Script configuration that are executed at certain events in the ActiveX control

The following sections describe configuration of these interfaces.

Although the Configuration dialog displays the list of all properties, methods and events, only the items that are required for the application need to be configured.

The screenshots used in the following sections depict the "Acrobat 3D Office" ActiveX control.

The names of the properties, methods and events vary for each ActiveX control, but the configuration interface is the same. The procedure described in the following is the same for all controls.

Configuring properties

The "Properties" tab contains a table with one line for each parametrizable property:

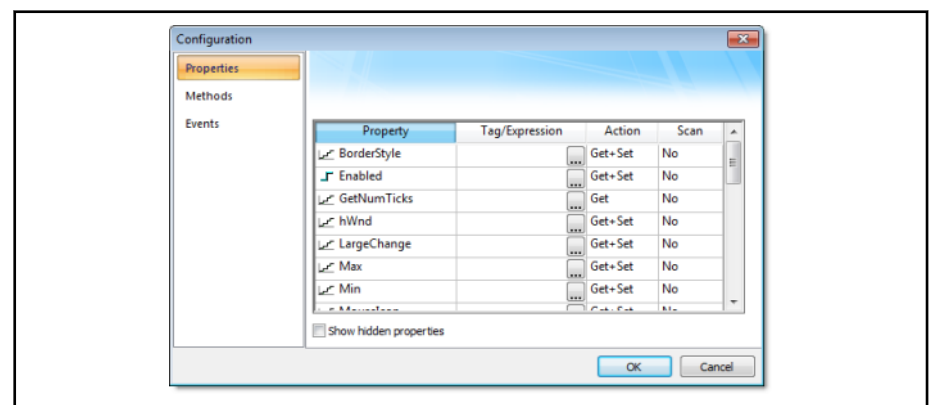


Fig. 8-67: Configuration dialog – Properties tab

- **Property** column: Description of all properties available in this ActiveX control including the respective type ID

Property symbol	Type
	Boolean
	Integer
	Real
	String

Screens and Graphics

- **Tag/Expression** column: The tag configured in this field is associated with the respective property of the ActiveX control. The "Action" column defines whether the value of this tag is written to the ActiveX property or if the value of the ActiveX property is written to this tag (or both).



In this field, an expression can be configured if the result of an expression is to be written to the property of the ActiveX object.

However, in this case, the value of the property cannot be read back to one tag (unless the XGet() function is used).

- **Action** column: In this column it is defined whether the parameter in the "Tag/Expression" column is read from the control or written to the control. In case that the value is to be read and written (Get+Set), the order of the actions has to be set (refer to the following table):

The "Action" column is automatically configured to "Set" if the expression in the "Tag/Expression" column cannot be written to.

Action	Description
Get	Reads the value of the ActiveX property and writes it to the tag configured in the Tag/Expression field
Set	Writes the value from the tag or expression configured in the Tag/Expression field into the ActiveX property
Get+Set	Executes both actions (Get and Set). However, when opening a screen with the ActiveX control, "WinStudio" executes the command Get before executing potential Set commands. This means, if "WinStudio" opens the screen in which the ActiveX control is configured, the tag configured in the Tag/Expression field is updated with the value from the ActiveX property
Set+Get	Executes both actions (Get and Set). However, when opening a screen with the ActiveX object, "WinStudio" executes the command Set before executing potential Get commands. This means, if "WinStudio" opens the screen in which the ActiveX element is configured the ActiveX property is updated with the value of the tag configured in the Tag/Expression field

- **Scan** column: Defines the polling method to get values from the ActiveX properties according to the following table:

Scan	Description
No	The value of the ActiveX property is only read and written to the tag configured in the Tag/Expression field if the screen with the ActiveX object is open and if the ActiveX object sends a message to "WinStudio" to update this tag
Always	"WinStudio" continuously reads the value of the ActiveX property and updates the tag configured in the "Tag/Expression" field with this value



Some ActiveX controls automatically send a message to the container (e.g. to the runtime environment) to indicate that a value has changed and can be read (Get).

However, some ActiveX controls do not support sending messages if values are changed.

In this case, the only way to get the updated values of the ActiveX properties is to keep calling these values from the ActiveX control (Scan=Always).

Configuring methods

The Methods tab provides a table with the following fields:

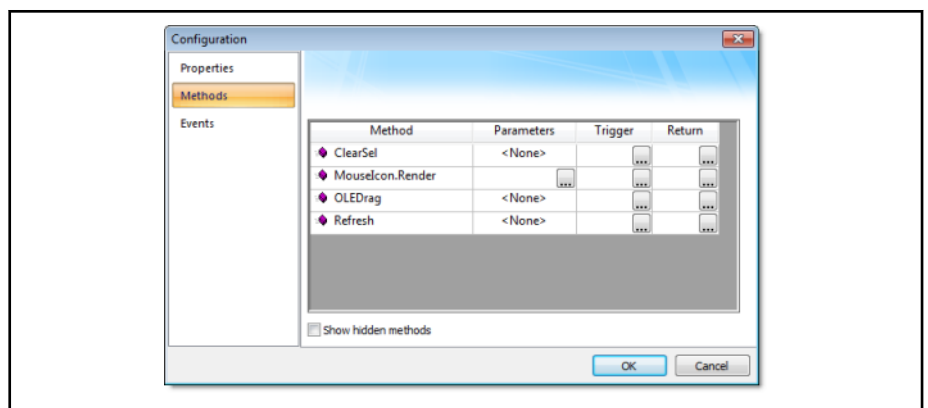


Fig. 8-68: Configuration dialog – Methods tab

- **Method** column: Lists all methods available from the ActiveX object
- **Parameters** column: The tags configured in this column are linked to the transfer parameters of a method.

If a method does not support any parameters, the text <None> is displayed in the Parameters field.

If the method expects more than one transfer parameter, a tag separated by a comma (",") can be entered for each parameter.

For example, **TagA , TagB , TagC**.

If the method is executed, either the value of the tags are written to the parameters of the method (input parameters), or the ActiveX writes the value of the parameters to the tags (output parameters) after the method is executed.



On clicking the Browse button (⋮), a list of parameters supported by the method is displayed to associate one tag with each parameter.

- **Trigger** column: If the tag configured in this column changes its value, the respective method of the ActiveX control is executed
- **Return** column: The tag configured in this column receives the return value of the method (if available)

Configuring events

The Events tab provides a table with the following fields:

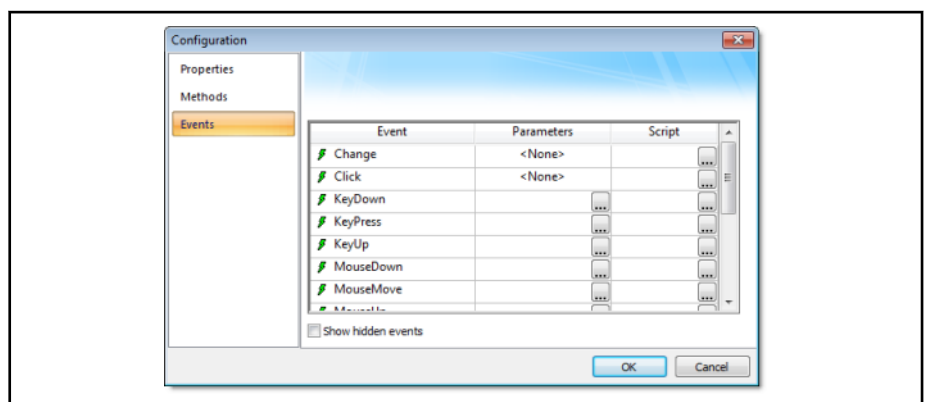


Fig. 8-69: Configuration dialog – Events tab

- **Event** column: Lists all events available from the ActiveX object

Screens and Graphics

- **Parameters** column: The tags configured in this column are linked to the transfer parameters of an event

If an event does not support any parameters, the text <None> is displayed in the Parameters field.

If the event expects more than one transfer parameter, a tag separated by a comma (",") can be entered for each parameter

For example, **TagA , TagB , TagC**.

If the event occurs, either the values of the tags are written to the parameters of the event (input parameters) or the parameter values are written to the tags (output parameters).



On clicking the Browse button () , a list of parameters that can be associated with a tag and are supported by the event is displayed.

- **Script** column: The script configured in this field is executed if the event is triggered by the ActiveX control



Click the Browse button () to open the program editor. The program can be created in VBScript or in the "Rexroth WinStudio" programming language

.NET Controls

.NET components are designed according to the Microsoft .NET framework, which is a standard for modular programming technologies. As WinStudio is a .NET container, .NET components can be integrated and executed in WinStudio screens. The actual functions of .NET components are contained in a **.NET control**, which provides the configuration dialogs.

.NET components include the following interfaces:

- **Properties:** Variables whose values can be read and/or written to the project (e.g. object color, file name, URL, etc.)
- **Methods:** Functions from the .NET component that can be triggered by the project (e.g., open dialog, execute calculation, etc.)
- **Events:** Internal messages that can trigger the execution of expressions in the project (e.g. Mouse_Click, Download_Completed, etc.)

.NET controls can differentiate in representation and function, depending on the configured properties, methods or events.

NOTICE

If .NET components are used in the project, make sure that the target system (runtime station) can support the same components and that they are properly installed and registered.

Selecting and placing .NET control

To select and insert .NET controls into WinStudio screens, proceed as follows:

1. In the **WinStudio Screen Tools** tool window, click on **.NET control**

Screens and Graphics

2. If the ".NET framework components" dialog opens (as in the following figure), it contains a list of all .NET components that are registered on the computer

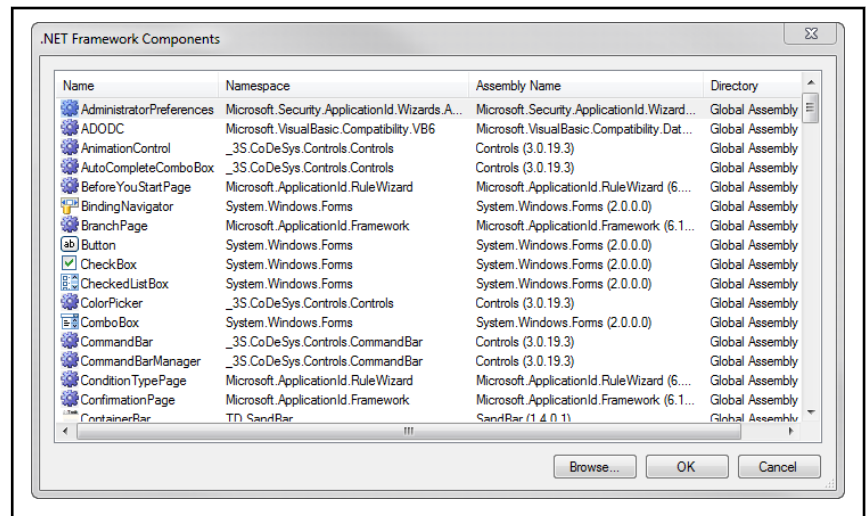


Fig.8-70: .NET framework components dialog

3. Select a component from the list, and click **OK** to place it in the screen. Click the **Browse...** button to find unregistered components on the computer. / Click **Browse...**



Registered .NET components are typically stored in the following directory:

C:\WINDOWS\Microsoft.NET\Framework

Unregistered .NET controls can be added to the selection list in the ".NET framework components" dialog by adding the following entry to the "projekt_name.APP" file:

[Execution Environment]

DotNetControlPath=OptionalPath

For example:

[Execution Environment]

DotNetControlPath=C:\DOTNET CONTROLS BACKUP

Thereafter, the ".NET framework components" dialog lists all registered components "and" all components found in the specified directory

4. By default, a new .NET control object is placed in the upper-left corner of the screen. Click on the object and drag it to the desired location
5. Once the object is placed, double-click on it to open its "Object Properties" dialog

Screens and Graphics

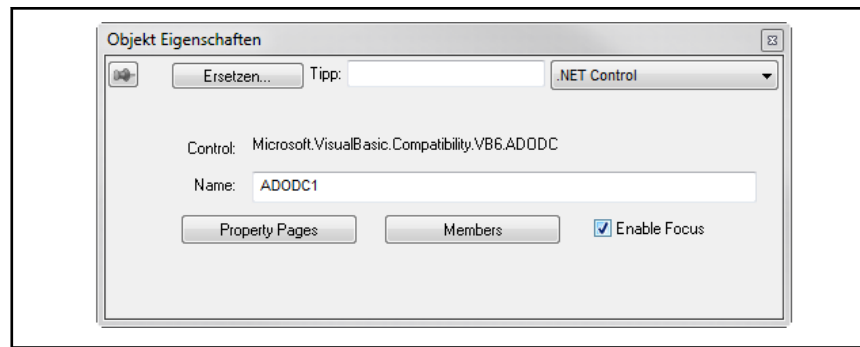


Fig.8-71: Object Properties: .NET Control

The "Object Properties" dialog shows the name of the .NET component. A unique name (alias) has to be assigned to the component in the **Name** field (e.g. CheckBox1). This name is used to reference the component when using the scripting languages ([VBScript on page 715](#) and [Integrated Script Language on page 471](#)).



.NET controls can never be configured with the same name in one WinStudio screen.

The **Property Pages** button opens the standard window for configuring the static properties (if available). The layout and the options in this dialog depend on the implementation of each .NET component. Use this interface to set properties that are not to be changed during runtime (fixed properties).

The **Members** button on the "Object Properties" dialog opens additional dialogs that enable the following tasks:

- Associate tags to properties of the .NET component
- Trigger methods from the .NET component by means of a tag change
- Configuration of scripts that are to be executed if events from the .NET component occur

The following sections describe configuration of these interfaces.



Although the "Members" dialog displays a list of all properties, methods and events, only the items that are required for the respective application need to be configured.

The following figures illustrate the .NET control "checkbox". Although the names of properties, methods and events varies by .NET control, the configuration interface is the same for any .NET component. The elements (properties, methods, events) described here apply for all .NET controls.

Configuring properties

The "Properties" tab provides a table with the following fields:

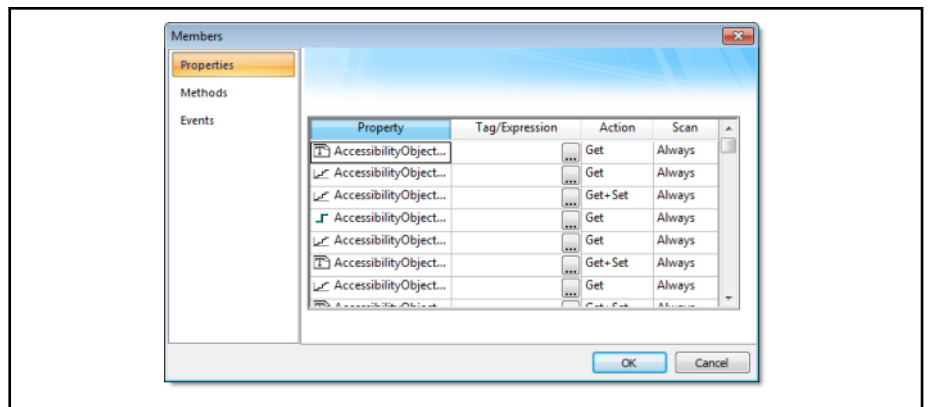


Fig. 8-72: Members dialog - Properties tab

- **Property** column: Lists all properties available from the .NET component, and indicates their types:

Property symbol	Type
	Boolean
	Integer
	Real
	String

- **Tag/Expression** column: The tag configured in this field is associated with the respective property of the .NET component. The Action column defines whether the value of this tag is written to the property or if the value of the property is written to this tag (or both)
- **Action** column: In this column, it is defined whether the parameter in the "Tag/Expression" column is read from the control or written to the control according to the following table:



If the value of the property is "read-only" (cannot be overwritten by the project), the **Action** field is automatically set to **Get**.

Action	Description
Get	Reads the value of the property and writes it to the tag configured in the Tag/Expression field
Set	Writes the value from the tag or expression configured in the Tag/Expression field to the property
Get+Set	Executes both actions (Get and Set). However: When opening a screen with the .NET component, WinStudio executes the command Get before executing potential Set commands. This means, if "WinStudio" opens the screen in which the .NET component is configured the tag configured in the Tag/Expression field is updated with the value from the property
Set+Get	Executes both actions (Get and Set). However: When opening a screen with the .NET component, WinStudio executes the command Set before executing potential Get commands. This means, if "WinStudio" opens the screen in which the .NET component is configured, the property is updated with the value of the Tag/Expression field

- **Scan** column: Defines the polling method to get values from the properties. For .NET components, all properties scan **Always** by default. "WinStudio" continuously gets the value of the property and updates the tag configured in the **Tag/Expression** field with this value

Screens and Graphics

Configuring methods The "Methods" tab provides a table with the following fields:

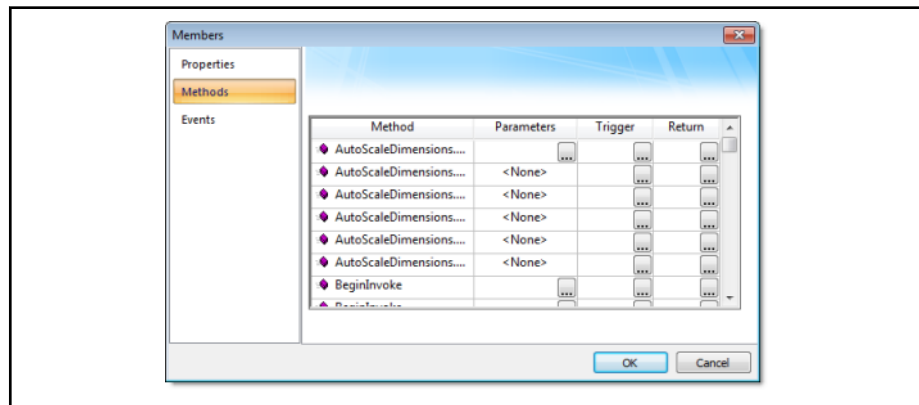


Fig.8-73: Members dialog – Methods tab

- **Method** column: Lists all methods available from the .NET component
- **Parameters** column: The tags configured in this column are linked to the transfer parameters of a method. If a method does not support any transfer parameters, the text <None> is displayed in the Parameters field. If a method expects more than one transfer parameter, a tag separated by a comma (",") can be entered for each parameter

For example, **TagA** , **TagB** , **TagC**.

On clicking the Browse button (...), a list of parameters supported by the method is displayed to associate one tag with each parameter.

If the method is executed, either the value of the tags is written to the transfer parameters of the method (input parameters), or the .NET control writes the value of the transfer parameters to the tags (output parameters) after the method is executed.

- **Trigger** column: If the tag configured in this field changes its value, the respective method of the .NET control is executed
- **Return** column: The tag configured in this field receives the return value of the method (if available)

Configuring events The "Events" tab provides a table with the following fields:

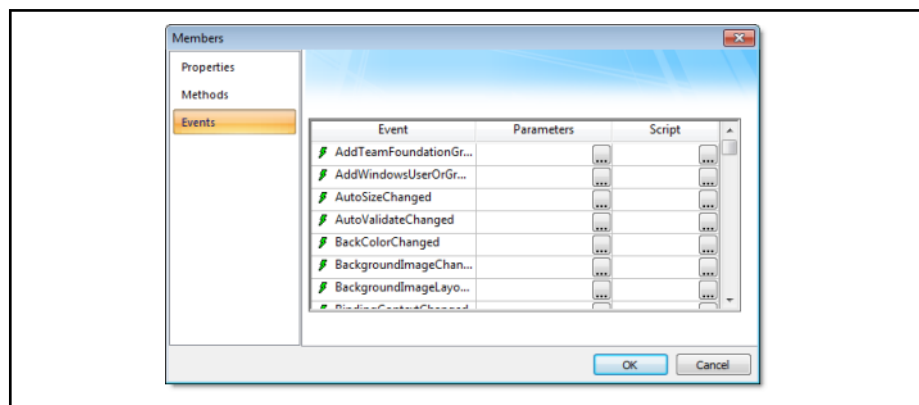


Fig.8-74: Members dialog - Properties tab

- **Event** column: Lists all events available from the .NET control
- **Parameters** column: The tags configured in this column are linked to the transfer parameters of events. If an event does not support any transfer parameters, the text <None> is displayed in the Parameters field. If the

Screens and Graphics

event expects more than one transfer parameter, a tag separated by a comma (",") can be entered for each parameter

For example, **TagA , TagB , TagC**.



On clicking the Browse button () , a list of parameters that can be associated with a tag and are supported by the event is displayed.

If the event occurs, either the value of the tags is written to the transfer parameters of the method (input parameters), or the .NET control writes the value of the transfer parameters to the tags (output parameters) after the event occurs.

- **Script** column: The script configured in this field is executed if the event is triggered by the .NET control



Click the Browse button () to open the program editor. The program can be created in VBScript or in the "Rexroth WinStudio" programming language

Adding a Link to an External Graphic File

By means of the library object "Graphic", external graphics files can be inserted as links into a WinStudio screen. This makes it possible to change the screen during runtime.

The graphics file can be stored in the project or at any other location:

- If the graphics file is to be downloaded to the target system with the other project files, it has to be stored in the project folder, e.g.
[...]\Projekt1\VEP40\user\config\
- If the file is stored at a different location on the target system or in the network, the absolute directory of the link is automatically saved



If the graphic is to be implemented directly into the WinStudio screen instead of a link to an external file and if the screen does not need to be changed during runtime, the screen object [Bitmap on page 173](#) can be used instead.

1. Click on **Graphic** in the **WinStudio Screen Tools** tool window
An "Open" dialog is displayed
2. Use this dialog to browse for and select the graphics file and click on **OK**
The graphic is inserted into the WinStudio screen as link
3. Double-click on the graphic object
The "Object Properties: Linked Picture" dialog is displayed

Screens and Graphics

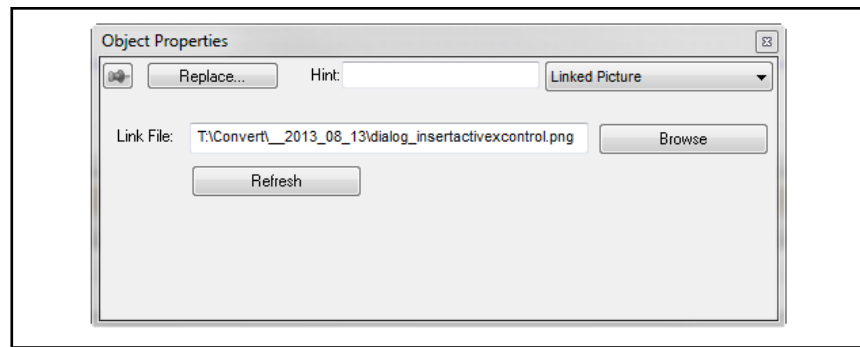


Fig.8-75: Object Properties: Linked Picture

4. Check the link to the graphic file in the **Link File** field

If the graphics file is stored in the project folder, the link is relative to this folder. If the graphics file is stored elsewhere, the link is a complete file path

5. If the link is to be changed during project runtime, enter a string tag as link (e.g. {tagname}). Then, during project runtime, the link is refreshed whenever the value of the string tag is changed

The tag value requires the same format as a normal link: A relative file path for a file stored in the project folder or an absolute file path for a file stored elsewhere on the target system or the network

6. Close the "Object Properties" dialog

8.3.5 Applying Animations to Screen Objects

Applying Animations to Screen Objects, General Information

Use the "WinStudio Dynamic Functions" toolbar to apply animations to a screen object or an object group. By means of animations, tags can be used to change object properties during runtime. Some animations also enable execution of commands or inserting values (command values) to the tags.



Fig.8-76: "WinStudio Dynamic Functions" toolbar

Dynamic objects are only displayed as active (in color) if the selected screen object or screen group supports this animation (e.g. text input / output for text fields). Move the cursor over an active WinStudio dynamic object to display the function in plain text.

Command Animation

Icon:

In the **WinStudio Dynamic Functions** toolbar, click on **Command** to add animations to a selected object or object group. This dynamic function enables triggering of actions or execution of program functions by mouse click during runtime. Double-click on the object to view its object properties.

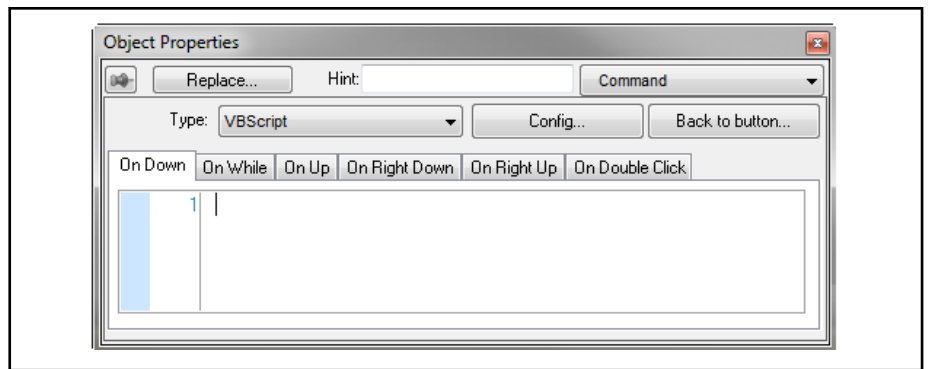


Fig. 8-77: Object Properties: Command

The configuration dialog provides a separate tab for each event supported by the object. Please note that more than one event can be configured simultaneously for the same dynamic function:

- **On Down:** Executes the command/script once when the object is clicked with the left mouse button
- **On While:** Executes the command/script continuously while the mouse button is pressed. The execution period (in milliseconds) for the command/script is defined in the "Rate" field. An exception is the option VBScript that is executed as quickly as possible
- **On Up:** Executes the command/script once when object is released by the left mouse button
- **On Right Down:** Executes the command/script once when the object is clicked with the right mouse button
- **On Right Up:** Executes the command/script once when the object is released by the right mouse button
- **On Double Click:** Use this item to execute the command/script once by double-clicking on the object with the left mouse button



- The runtime project handles touchscreen actions the same as mouse cursor actions
- The events "On Right Down", "On Right Up" and "On Double Click" are not supported for projects that are executed on target systems with Windows CE
- When creating a project for touchscreen devices, keep in mind that events "On Right Down" and "On Right Up" cannot be triggered on such devices

- **Type selection field:** This setting defines the type of action that is to be executed if the event occurs. Please observe that for each event (e.g. on pressing or releasing) an individual type selection can be made. This enables different configuration of the same dynamic functions for different events. The following types are supported:

Screens and Graphics

Type	Description
Built-in Language	Enables configuration of a script using the "WinStudio" integrated language. If this type is selected, up to 12 expressions for each event can be configured in the Expression column. The expressions are executed sequentially from the first line to the last one if the event is triggered. The result of each expression is written to the tag configured in the Tag column (if any). For further information, refer to Appendix: WinStudio-Internal Scripting Language on page 471
VBScript	Enables configuration of a script using the standard VBScript language. If this type is selected, a script can be configured in the VBScript editor for the command animation . For further information on VBScript language, refer to Overview of VBScript on page 715
Open screen	Enables configuration of the command animation to open a specific screen if the event is triggered during runtime. This type is equivalent to the function Open on page 529 . Enter the screen name into the Open screen field or select a screen via the Browse dialog. A string tag can also be entered in this field between curly brackets {TagName}. If the event is executed, the project attempts to open the named screen
Close screen	Enables configuration of the command animation to close a specific screen if the event is triggered during runtime. This type is equivalent to the function Close on page 528 . Enter the screen name into the Close screen field or select a screen via the browser dialog. A string tag can also be entered in this field between curly brackets {TagName}. If the event is executed, the project attempts to close the named screen
Set variable	Enables configuration of the command animation to set a tag if the event is triggered during runtime. Enter the tag name into the Set variable field or select a tag via the Browse dialog. If the event is executed, the project writes the value 1 to the configured variable
Reset variable	Enables configuration of the command animation to reset a variable if the event is triggered during runtime. Enter the tag name into the Reset variable field or select a tag via the Browse dialog. If the event is executed, the project writes the value 0 to the configured variable
Toggle variable	Enables configuration of the command animation to toggle a variable if the event is triggered during runtime. Enter the tag name into the Toggle variable field or select a tag via the Browse dialog. If the event is executed, the project toggles the value of the tag configured in this field

- **Config** button: Opens an extended view of the object properties with additional configuration settings
- **Back to** button: Click to return to the object properties of the underlying button object

Configuration dialog Extended object properties dialog with additional configuration settings...

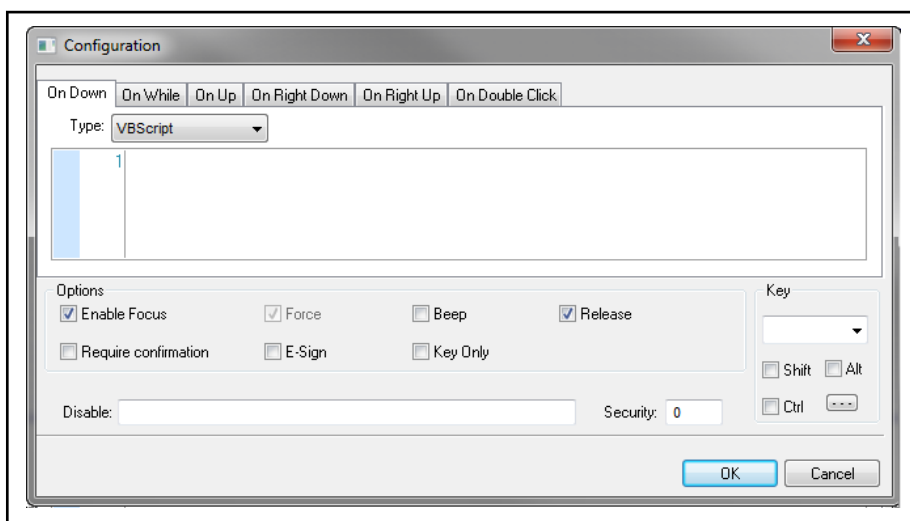


Fig.8-78: Configuration dialog

Screens and Graphics

The event tabs (e.g. "On Down", "On While", etc.) and the **Type** selection list are the same as in the "Object Properties" dialog described above. The additional configuration settings are shared for all events:

- "Options" pane:
 - **Enable Focus:** If this option is selected, the object on which the command animation has been executed, can receive the input focus during runtime and selected by navigating with the <tab> key
 - **Force:** If this option is activated, all values assigned to variables trigger change events, even if the value of the respective tag does not change. For instance, if a tag has the value 0 and the command animation writes the same value 0 to this tag, any other task in the runtime project recognizes that this tag changed its value (even if it did not) after executing the animation. This option is useful to make sure that actions driven by tag changes (e.g. write on tag change from a communication driver) are triggered after the command animation is executed

For projects created with "Rexroth WinStudio v7.x" or earlier, the **Force** option is enabled by default and cannot be disabled
 - **Beep:** If this option is enabled, a short beep is returned when the command is executed. This option is useful to provide an audio feed-back indicating that the command was executed. It does not indicate, however, if the action triggered by the command animation was successful or not
 - **Release:** If this option is enabled, the "On Up" event is executed when the cursor (or finger) is dragged out of the object area (whether the button was released or not). This option is useful to make sure that the "On Up" event is always executed after an "On Down" event, even if the mouse cursor is moved out of the object area before releasing it
 - **Require confirmation:** If this option is enabled, a confirmation is required before executing the command. This option is useful for decreasing the accidental triggering of critical events during runtime
 - **E-Sign:** If this option is enabled, a prompt is displayed to enter the electronic signature before executing the command
 - **Key Only:** If this option is enabled, "only" the keyboard shortcuts (configured in the "Key" pane described below) can be used to execute commands
- **Disable:** Disables user action if the result of the expression configured in this field is TRUE (value different from 0)
- **Security:** [Security on page 357](#), an access level required to execute the command animation during runtime
- **Key** pane: Keyboard shortcut to trigger the events "On Down", "On While" and "On Up" using the keyboard. (In other words, pressing this keyboard shortcut is the same as clicking the left mouse button.) This option is particularly useful if the keyboard is the only available physical interface to interact with the user interface during runtime
 - **Shift, Ctrl or Alt:** Click to create a combination key, meaning the Shift, Ctrl and/or Alt key has to be pressed with the key specified in the drop-down list
 - Click the browse button (...) to open the "Key Modifier" dialog, to modify the combination keys. **Left, Right or Left or Right** can be se-

Screens and Graphics

lected to specify the position on the keyboard of the Shift, Ctrl or Alt key in the key combination. If **Left or Right** is selected, the command is executed any time one of these keys is pressed in combination with the key specified in the selection list

Hyperlink Animation



Icon:

In the **WinStudio Dynamic Functions** toolbar, click on **Hyperlink** to add a hyperlink to a selected object or object group.

Hyperlinks enable opening a standard browser with a predefined URL during runtime.

Double-click on the object to open the "Object Properties" dialog.

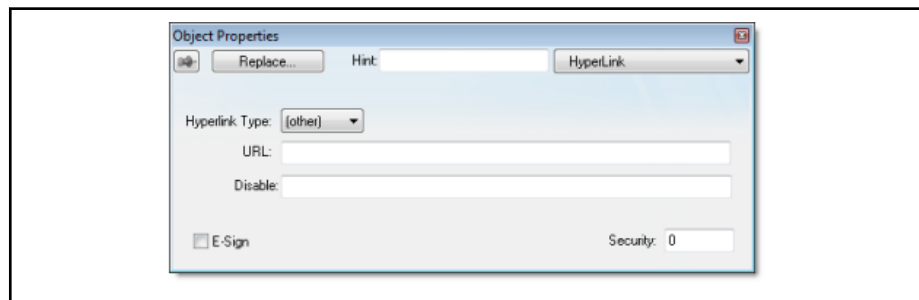


Fig.8-79: Object Properties: Hyperlink

Use this dialog to specify the following parameters:

- **Hyperlink Type:** Select a URL protocol from the selection list. The project uses this protocol when loading a URL
- **URL:** Enter the URL address to be loaded



It is not necessary to enter the protocol type in the URL field.

When selecting a protocol type from the **Hyperlink Type** list, the project automatically adds the prefix of the protocol to the URL address.

- **Disable:** Enter a value greater than zero into this field to disable the hyperlink command animation for the selected object(s)
- **E-Sign:** If this option is enabled, a prompt is displayed to enter the electronic signature before executing the animation
- **Security:** Enter a value into this field to specify a security level for the object(s) as defined under [Project Security on page 357](#). If a user logs on but does not have the required security level, the project disables the hyperlink command for the object(s)

Bargraph Animation



Icon:

In the **WinStudio Dynamic Functions** toolbar, click on **BarGraph** to add a bargraph property to a selected object or object group. Double-click on the object to open its "Object Properties" dialog.

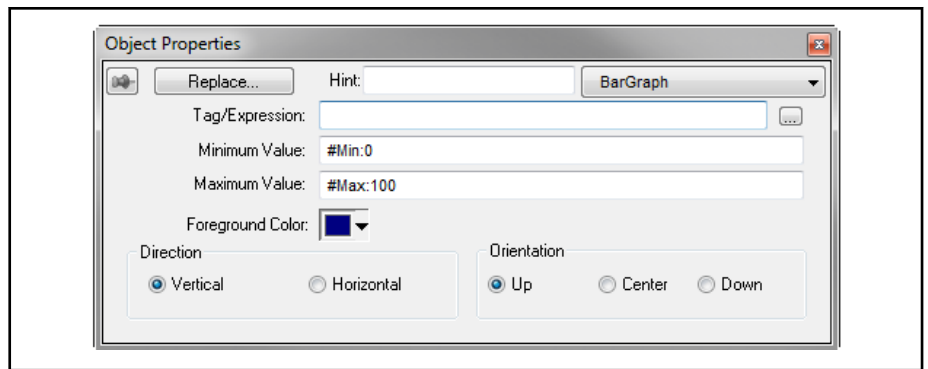


Fig. 8-80: Object Properties: BarGraph

Use the "Object Properties" dialog to specify the following parameters:

- **Tag/Expression:** Enter a tag or expression that is to be displayed as bar height or width. Click on the  button to browse for existing tags or expressions
- **Minimum Value:** Enter a numeric constant or a tag value into this field to define the minimum value used to calculate the height (if vertical) or width (if horizontal) of the bars
- **Maximum Value:** Enter a numeric constant or a tag value into this field to define the maximum value used to calculate the height (if vertical) or width (if horizontal) of the bars

If no value is specified for this field, the application opens a dialog for selecting or creating a tag



"WinStudio" also allows to enter constants in tag/numeric value fields. Constant values (defined by the # character) are equivalent to numeric values. The text constant can be used as description field as well as to specify generic values.

For example: **#Name:100**.

Where the value ("100") following the colon (":") is the generic value and "Name" is only a mnemonic for the constant and is not added to the database.

- **Foreground color:** To specify a filling color for the bar, click on the color box
- **Direction:** Click the **Vertical** or **Horizontal** radio button to specify the direction of the bar graph
- **Orientation:** Click the **Up**, **Center**, or **Down** radio button to specify the orientation of the maximum and minimum values when drawing the bar

Text Data Link Animation



Icon:

Click on **Text Data Link** in the WinStudio dynamic function toolbar to add the properties for input or output texts for the animation to a selected [text object on page 175](#).

The dynamic function "Text Data Link" enables representation and change of tag values via the user interface during runtime.

Screens and Graphics



This animation can only be applied to text objects that include one or more **#** characters.

Each **#** represents an input or output character. **#** characters can be combined with regular text in the same text object, for example, **MyLabel#####** or **\$###.##**.

It is important to remember that the runtime project always displays the highest digits of a numeric value, regardless of the number or placement of **#** characters in the text. That means, if not sufficient **#** characters are available to display the value, then it is transformed depending on the format of the value (as set by the **Fmt** option described below):

- In decimal format, the number of decimal places is determined by the position of the decimal separator in the **###** text. However, if not sufficient **#** characters are provided to the left of the decimal separator to display the whole value, the integer overwrites the decimal value

For example, if a value of 112.64 is to be displayed in **###**, only "112" is displayed

- In hex and binary formats, if more **#** characters are provided than are required to display the value for Hex and Binary formats, the runtime project fills these in with leading zeroes. If less characters than required are available, the value simply is shortened
- In Auto format, the runtime project ignores the number of **#** characters and displays the entire numeric or string value

Decimal values are displayed with all pre-decimal and decimal places, regardless of the position of the decimal separator in the **###** text. Exceptionally large values or long strings may disrupt the layout of the screens.

Double-click on the object to open the "Object Properties" dialog. Use this dialog to specify the following parameters:

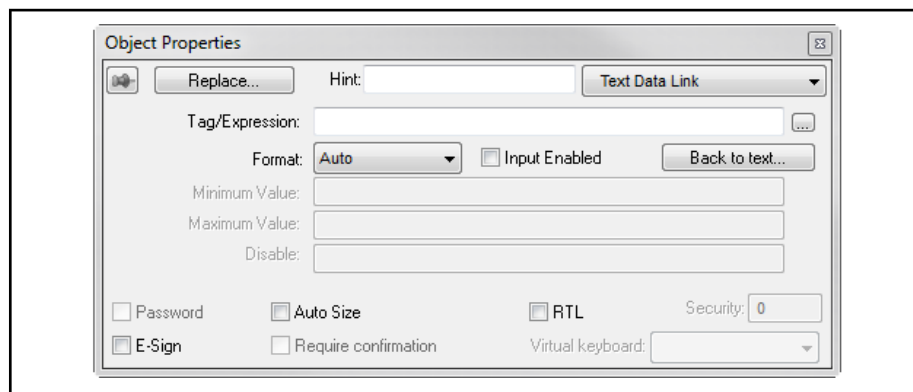


Fig.8-81: Object Properties: Text Data Link

- **Tag/Expression**

Enter the following into the text field:

- Name of a tag whose value is to be displayed or entered
- Expression whose values are to be displayed

Open the "Object Finder" with the  button to select an existing tag or expression.



If the configured tag/expression is invalid, the placeholder characters (###) are displayed during runtime.

- **Format:** Selection of how a numeric value of the specified tag or expression is to be formatted and displayed on the screen. Available options include **Decimal**, **Hexadecimal**, **Binary** and **Auto**. If **Auto** is selected, the value is formatted according to the virtual table created by the [SetDecimalPoints on page 604](#) function. The format settings do not change the storage format of the displayed tag or expression
For example: **Tag/Expression** is set to a tag of [Integer on page 113](#) type, **Input Enabled** is checked and **Fmt** is set to **Hexadecimal**. A new value can be entered in hexadecimal format but it is saved in the project database as Integer
- **Input Enabled:** Click (enable) this option to allow user input for the specified tag. Disable (deactivate) this option if the value is only to be displayed at runtime without changing it
- **Back to text:** Click to return to the object properties of the underlying text object
- **Minimum Value:** Enter a minimum value for the tag associated with this text object. It is not possible to enter lower values
- **Maximum Value:** Enter a maximum value for the tag associated with this text object. It is not possible to enter higher values
- **Password:** Enable (activate) this option if the input field is used for a password input, for example, and all characters are to be replaced by a "*****"
- **Require confirmation:** Enable (activate) this option to require users to confirm any new values set during runtime
- **Auto Size:** Enable (activate) this option to automatically resize the text object to fit the text length. This option is not available if **Input Enabled** is enabled (see above)
- **E-Sign:** If this option is enabled, a prompt is displayed to enter the electronic signature before changing the tag value
- **Virtual keyboard:** For this object, a virtual keyboard is used to enter values. Before configuration of the virtual keyboard for this interface, the option "virtual keyboard" in the "Viewer" tab in the properties of the "WinStudio" folder is to be activated
- **Disable:** Enter a value above zero in this field to deactivate the input function
- **Security** Enter a value into this field to specify the security level that is required to edit a value at runtime. (also refer to [Security on page 357](#))

Color Animation



Icon:

In the **WinStudio Dynamic Functions** toolbar, click on **Color change** to add the animation to a selected object.

The color animation enables modification of the color of an object during runtime based on the value of a tag or expression.

Double-click on the object to open the Object Properties dialog.

Screens and Graphics

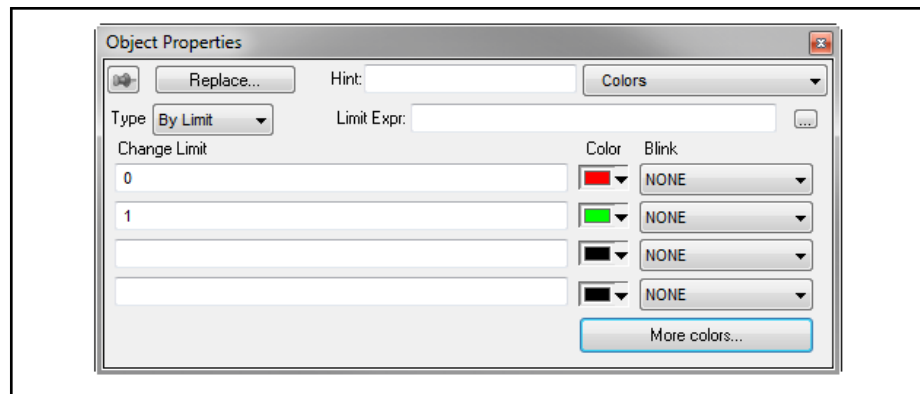


Fig. 8-82: Object Properties: Colors

Use this dialog to specify the following parameters:

- **Type:** Specifies the mode of the color change:
 - **By Limit:** If this type is selected, up to four limit values (Change Limit) and a color for each limit value can be defined. If the value of the tag or expression configured in the Tag/Expression field reaches the limit value, the color associated with the respective limit value is applied to the object
 - **By Color:** If this type is selected, any color code to be displayed is provided to a tag in the field Tag/Expression. Using this code, any color supported by the device can be applied to the object



The function [RGB Color on page 601](#) can be configured in the Tag/Expression field if the entry "By Color" is selected under **Type**. This enables configuration of colors by its RGB codes. Refer to [Color Interface on page 241](#) for a table with the codes for the most commonly used colors.

- **Tag/Expression:** Enter the name of a tag or expression to be monitored. If the "By Limit" type is selected, "WinStudio" compares the result of the tag/expression with the defined limit values to determine the correct color for the selected object. If "By Color" is selected, the value of the tag parameterized here defines the color of the object (color code)
- **Change Limit:** Enter a limit value (a numeric constant or tag) for the color change. The numbers have to be configured in ascending order (from top to bottom). Click on "More colors" to configure up to 16 different limit values for the color animation

Screens and Graphics

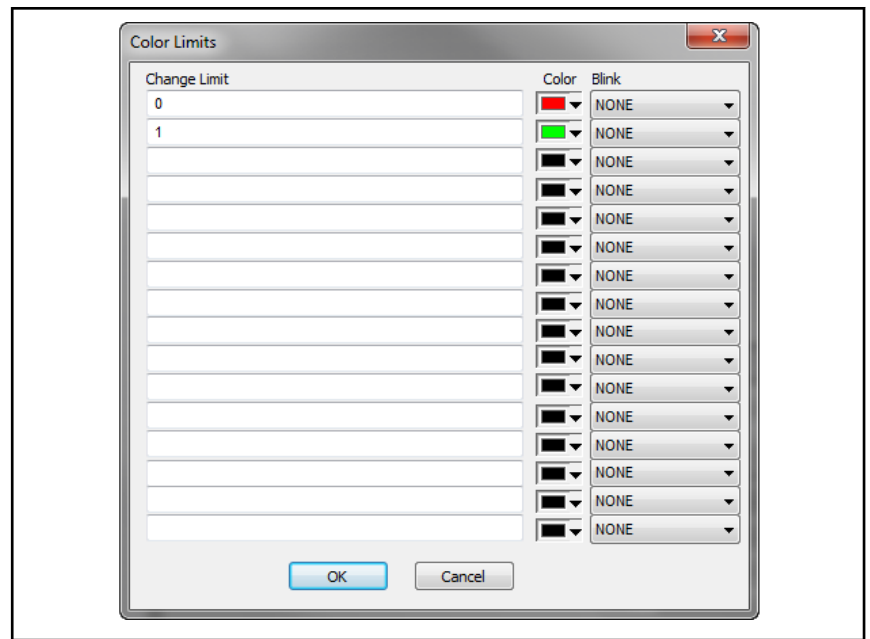


Fig. 8-83: Color Limits dialog

- **Color:** Click on the button to assign a color to each color change limit value
- **Blink:** Select whether the color is to blink on reaching the limit value and how fast



The following fields are automatically disabled (grayed out) if "By Color" is selected under **Type: Change Limit, Color and Blink.**

Position Animation



Icon:

The position animation enables moving an object horizontally and/or vertically at runtime.

In the **WinStudio Dynamic Functions** toolbar, click on **Position** to add the animation to a selected object.

Double-click on the object to open the "Object Properties" dialog.

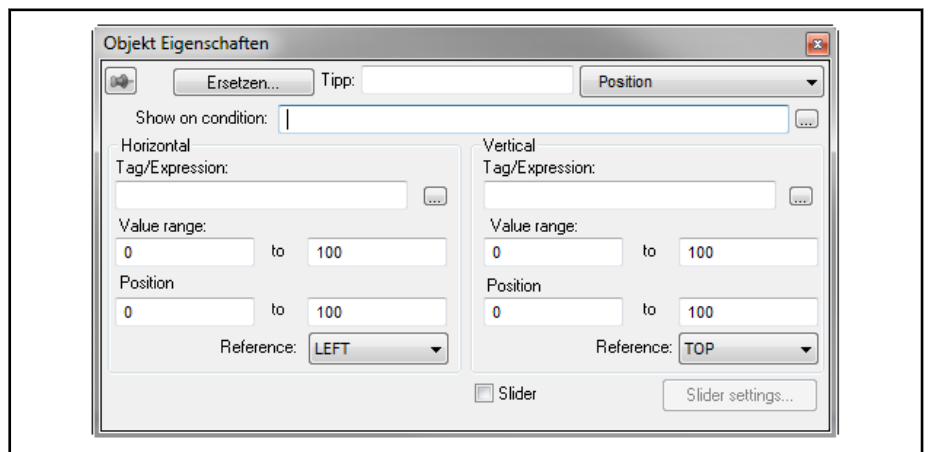


Fig. 8-84: Object Properties: Position

Screens and Graphics

Use the dialog to configure the following properties:

- **Show on condition:** Configure a tag/expression in this field to define when the object is shown. If the field is blank or the tag/expression is TRUE (i.e. all values different from zero), the object is shown. If the value is FALSE (i.e. 0), the object is not shown. Thus, the field can be left blank to always show the object
- **Slider:** Click (enable) this option to operate the object like a slider, which means that the object can be dragged around the screen to send values to the configured tags. (Refer to "Move" below.) Click on the **Slider settings** button to configure additional settings (activation/deactivation of access rights)
- "Horizontal" and "Vertical" panes: Configure these settings to determine how the object moves in the WinStudio screen:
 - **Tag / Expression:** The meaning of these fields depends on whether the **Slider** option is enabled (see above):
 - If the **Slider** option is not enabled, either a tag "or" an expression is to be configured in this field. The current value of the tag / expression determines the current position of the object. If the value changes, the object is moved on the screen
 - If the **Slider** option is enabled, only an Integer or Real tag is permitted in this field. The tag receives a value that is equivalent to the current position of the object. If the object is moved, the value changes

The position value for a "horizontal" movement increases if the object is moved to the right and decreases if the object is moved to the left. The position value for a "vertical" movement increases if the object is moved to the bottom and decreases if the object is moved to the top

- **Value range:** Enter the minimum and maximum values for the tag / expression. If the actual value exceeds this range, the value is ignored and the limit value is used instead
- **Position:** Enter values to specify how far (in pixels) the object can move from its starting position. The starting position is "0.0". Values greater than 0 allow the object to move right and down, and values less than 0 allow the object to move left and up. During runtime, the object's position is proportional to the tag / expression value within its scope. For example, if **Position** is "0" to "100" and **Range** is "0" to "10", then each increment in the value moves the object 10 pixels. This applies to horizontal and vertical movements
- **Reference**

Select a reference point on the object. The following table illustrates how the selections for "Horizontal" and "Vertical" work in combination:

	LEFT	CENTER	RIGHT
TOP			

Screens and Graphics

CENTER			
BOTTOM			

This **reference point** is only important if the [size animation on page 231](#) has been added to the same object. The position of the object is always based on this reference point, regardless of the size or shape of the object.

Resize Animation



Icon:

The "Resize" animation enables increasing or decreasing the size of an object during runtime.

In the **WinStudio Dynamic Functions** toolbar, click on **Resize** to add the animation to a selected object.

Double-click on the object to open the "Object Properties" dialog.

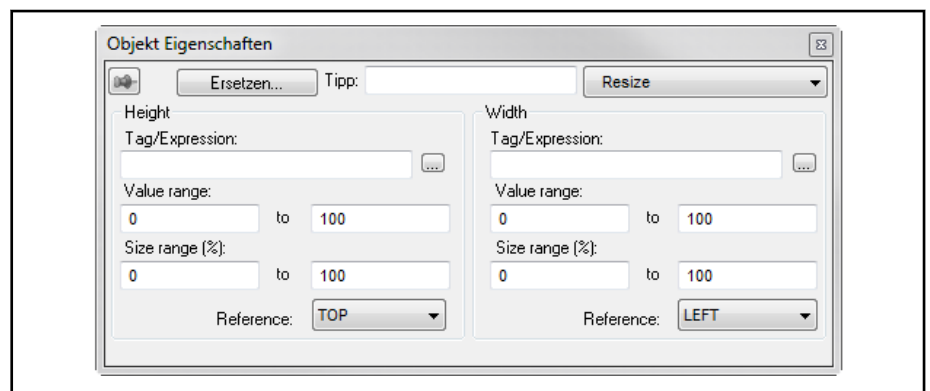


Fig. 8-85: Object Properties: Size

Use the dialog to configure the following properties:

- **Tag/Expression:** Enter the tags that define the **Height** and **Width** of the object. Leave one of the fields blank to prevent the object from changing its size in that dimension
- **Value range:** Enter the minimum and maximum values for the specified tag(s). If the actual value exceeds this range, the value is ignored and the limit value is used instead
- **Size range (%):** Enter the minimum and maximum values for the size of the object. The minimum value can be as low as 0% (making the object effectively invisible), and the maximum value can be as high as required. The original size of the object is 100% if it is drawn in the screen editor. Thus, 200% is twice its original size, etc.

During runtime, the size of the object is proportional to the tag value within its range. For example, if the **Size (%)** is "0" to "100" and the **Range** is "0" to "10", then each increment in the value increases the object size by 10%. This applies for both height and width

- **Reference:** Select a reference point to determine the directions in which the object changes its size. The following table illustrates how the selections for height and width work in combination:

Screens and Graphics

	LEFT	CENTER	RIGHT
TOP			
CENTER			
BOTTOM			

Rotation Animation



Icon:

In the **WinStudio Dynamic Functions** toolbar, click on **Rotation**, to add the animation to an object [Line on page 167](#), [Open Polygon on page 168](#), [Closed Polygon on page 169](#) or [Bitmap on page 173](#).

Double-click on the object to open the "Object Properties" dialog.

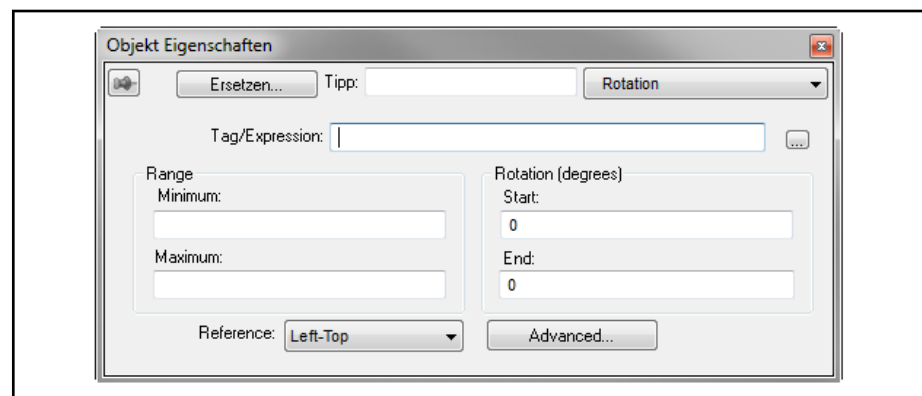


Fig.8-86: Object Properties: Rotation animation

Use this dialog to specify the following parameters.

- **Tag/Expression:** Enter a tag name or expression to activate the rotation animation. The value of **Tag/Expression** determines the actual rotation of the object. The degree of rotation changes with the value
- "Value range": Enter the **Minimum** and **Maximum** values allowed for **Tag/Expression**. Values less than the minimum and greater than the maximum are disregarded
- "Rotation (degree)": Enter the **Start** and **End** positions (in degrees) of the object. The actual rotation is proportional to the value of **Tag/Expression** within the "Range". An object can rotate up to 360 degrees and it rotates clockwise by default



For example, a rotation animation has the following settings: **Minimum** is 0, **Maximum** is 100, **Start** is 0 and **End** is 180. If the current value of **Tag/Expression** is 50 (i.e. halfway between Minimum and Maximum), the actual rotation of the object is 90 degrees (i.e. halfway between Start and End). A value of 25 is equal to 45 degrees, a value of 75 is equal to 135 degrees, etc.

- **Reference:** Select one of the following as a pivot point to rotate the object:
 - **Left-Top:** Upper-left corner of the object
 - **Left-Bottom:** Lower-left corner of the object
 - **Center:** Center of the object
 - **Right-Top:** Upper-right corner of the object
 - **Right-Bottom:** Lower-right corner of the object

The pivot point can be exactly positioned by means of an Offset (Advanced button) as described in the following

- **Advanced:** Click on the button to open the "Advanced" dialog to configure the following settings:

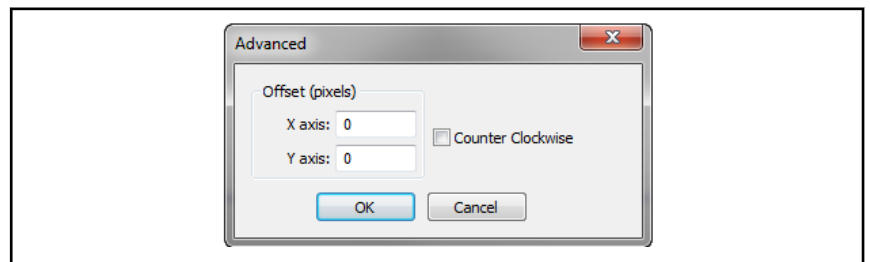


Fig. 8-87: Object Properties: Rotation animation – Advanced dialog

- "Offset (pixels)": Enter the number of pixels by which the **Reference** (i.e. pivot point) is to offset on the **X axis** and/or **Y axis**
- **Counter Clockwise:** Enable (activate) this option to make the object rotate counter-clockwise

8.4 Formatting Screen Objects

8.4.1 Formatting Screen Objects, General Information

The "WinStudio Alignment Functions" and "WinStudio Object Properties" toolbars offer numerous functions to align, group and design screen objects.



Fig. 8-88: "Alignment Functions" toolbar

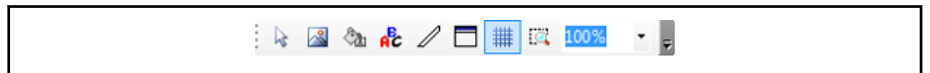


Fig. 8-89: "Object Properties" toolbar



Some functions in the toolbars are only available if one or multiple objects have been selected in the WinStudio screen.

Move the cursor over an active "tool" to display the function in plain text.

The tools are organized into the following groups:

- **Arrange:** Arranging objects in a WinStudio screen including
 - [Move to Foreground and Move to Background on page 234](#)
 - [To the Foreground and To the Background, by One Level on page 235](#)
 - [Group and Ungroup Objects on page 235](#)

Screens and Graphics

- [Align, Center and Distribute on page 236](#)
- [Rotating and Mirroring on page 238](#)
- [Adjusting the Position on page 239](#)
- **Size:** [Same Width / Same Height for Selected Objects on page 239](#) of a screen object in a WinStudio screen
- **Style:** Changing the color of screen objects
 - [Fill Color on page 240](#) of a screen object
 - [Line Color on page 241](#) of a screen object
- **Edit the color of a component:** [Color Interface on page 241](#)
- **Fonts:** Changing the [Fonts on page 244](#) of a screen object

8.4.2 "Move to Foreground" and "Move to Background"

"WinStudio" assigns a unique identification number (ID#) to every screen object. These ID#s always start at zero and range up to the total number of objects on the WinStudio screen.

Click on an object to display its ID# in the [Status Bar on page 87](#).

"WinStudio" uses ID#s to determine whether an object is displayed in front of or behind another screen object. Objects with lower ID#s are displayed behind objects with higher ID#s.

With the following object layer tools, selected objects can be moved behind or in front of other screen objects.

These tools can be used only with a single selected object or a [group on page 235](#) of objects. If several individual objects are selected and moved behind or in front of another object, the selected objects maintain their original display order.

Move object to the background

In the **WinStudio Alignment Functions** toolbar, click on , **Move object to the background** to move one or multiple selected objects behind all other objects in the screen. "WinStudio" assigns the object the lowest ID# and moves it behind all other objects on the screen.

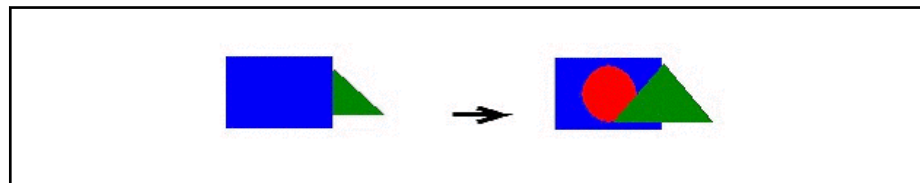


Fig. 8-90: Moving objects to the background

Alternatively, right-click on an object and select **Move to background** from the context menu of the object.

Move object to the foreground

Click on the  tool, **Move object to the foreground**, to move a selected object or objects in front of all other objects on the screen. "WinStudio" assigns the object the highest ID# and moves it in front of all other objects on the screen.

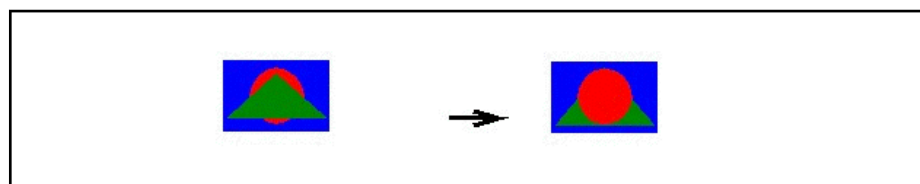


Fig. 8-91: Moving objects to the front

Alternatively, right-click on an object and select **Move to foreground** from the context menu.

8.4.3 "To the Foreground" and "To the Background", by One Level

"WinStudio" assigns a unique identification number (ID#) to every screen object. These ID#s always start at zero and range up to the total number of objects on the screen. Click on an object to display its ID# in the [Status Bar on page 87](#).

"WinStudio" uses ID#s to determine whether an object is displayed in front of or behind another object in the screen. Objects with lower ID#s are displayed behind objects with higher ID#s. With the following object layer tools, selected objects can be moved behind or in front of other screen objects.



These tools can only be used with a single selected object or [group on page 235](#) of objects. These tools cannot be used with multiple objects selected. If several individual objects are selected and moved behind or in front of another object, the selected objects maintain their original display order.

Open the context menu of a selected object and click on **To the back** to move the selected object/group of objects by one level behind the next object in the screen. "WinStudio" assigns the selected object the next available ID# less than the object behind which it was moved.

Open the context menu of a selected object and click on **To the front** to move the selected object/group of objects by one level in front of the next object in the screen. "WinStudio" assigns the selected object the first available ID# greater than the object in front of which it was moved.

8.4.4 "Group " and "Ungroup Objects"

Symbols of the "Alignment Functions" toolbar:

Use the following tools to group and ungroup two or more selected objects.



All objects with [animations on page 220](#) and groups of symbol objects (which includes most symbols and library objects) have multiple object and dynamic properties.

The selection list in the "Object Properties" dialog, provides access to these various dialogs and properties.

Grouping objects

In the **WinStudio Alignment Functions** toolbar, click on , **Group objects**, to combine several selected objects to a single object for easier handling. (Each part of the group can be accessed in the "Object Properties" dialog.)



Alternatively, right-click on an object and select **Group** from the context menu.

Ungrouping objects

In the **WinStudio Alignment Functions** toolbar, click on , **Ungroup**, to divide a grouped object into its individual components.



Alternatively, right-click on an object and select **Ungroup** from the context menu.

Screens and Graphics



A complex grouped object can consist of several sets of grouped objects ("subgroups"). Consequently, it may be necessary to ungroup all of the subgroups to completely ungroup a complex object.

8.4.5 "Align", "Center" and "Distribute"

If a series of objects (two or more) is selected, those objects can be aligned based on the location of the last object selected. The last object selected is assigned a special mark (filled check icon).



In all of the figures, the rectangle represents the last object selected.

Use the following tools from the **WinStudio Alignment Functions** toolbar to align a series of objects.

Examples are provided in the following figures:

Left-aligned

Click on , **Left-aligned**, to align all selected objects to the left edge of the last object selected.

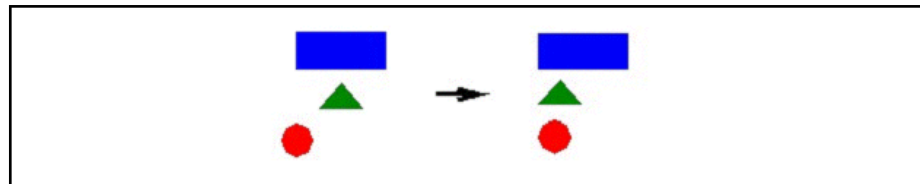


Fig. 8-92: Aligning objects left

Right-aligned

Click on , **Right-aligned**, to align all selected objects to the right edge of the last object selected.

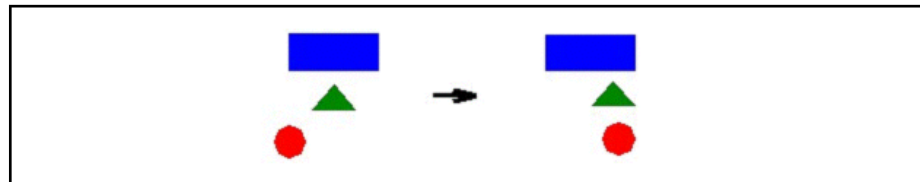


Fig. 8-93: Aligning objects right

Align top

Click on , **Align top**, to align all selected objects to the top edge of the last object selected.

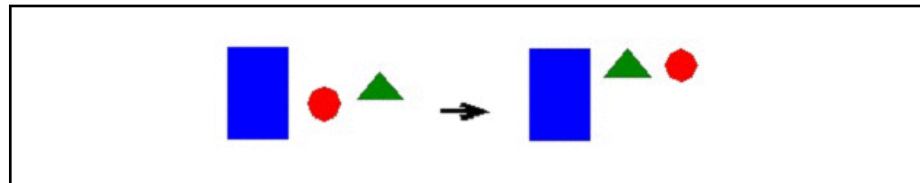


Fig. 8-94: Aligning object to the top

Align bottom

Click on , **Align bottom**, to align all selected objects to the bottom edge of the last object selected.

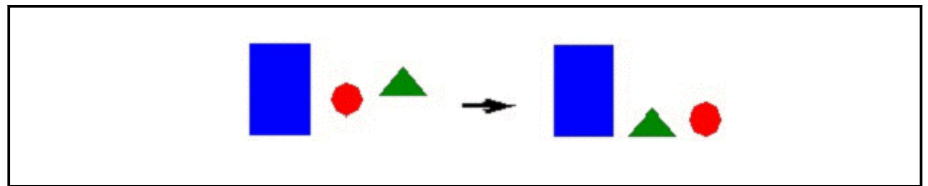


Fig. 8-95: *Aligning object to the bottom*

Center vertically

Click on , **Center vertically**, to align all selected objects to the vertical center of the last object selected.

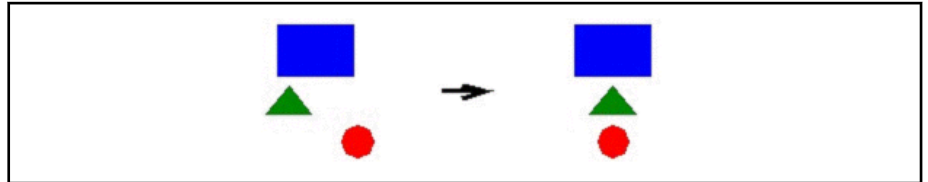


Fig. 8-96: *Centering objects vertically*

Centering horizontally

Click on , **Center horizontally** to align all selected objects to the horizontal center line of the last object selected.

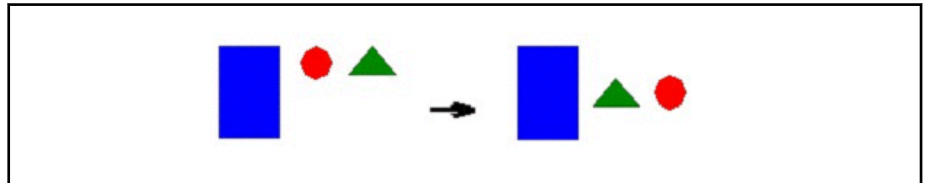


Fig. 8-97: *Centering objects horizontally*

Distributing horizontally

Click on , **Distributing horizontally** to put an equal amount of horizontal space between a series of objects (three or more).

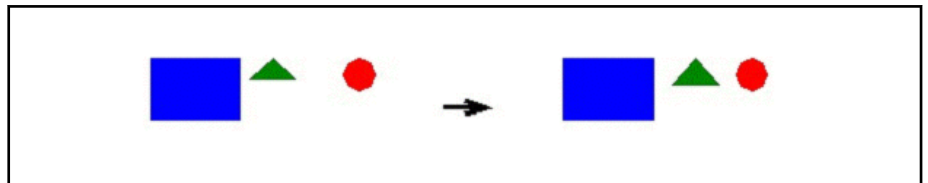


Fig. 8-98: *Distributing objects horizontally*

Distributing vertically

Click on , **Distributing vertically** to put an equal amount of vertical space between a series of objects (three or more).

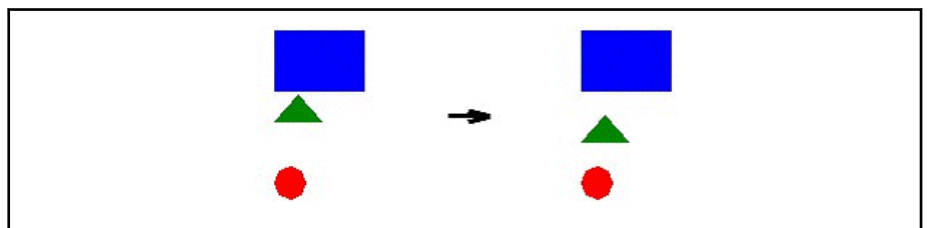


Fig. 8-99: *Distributing objects vertically*



The distribution tools may move the last object selected (with filled handles) by no more than a few pixels to equally distribute all of the objects.

Screens and Graphics

8.4.6 "Rotating" and "Mirroring"

"Rotating"

Use the following tools from the **WinStudio Alignment Functions** toolbar to rotate and mirror objects.

Rotate

Click on the tool , **Rotate**, to rotate the selected object by 90 degrees (a quarter turn) clockwise:

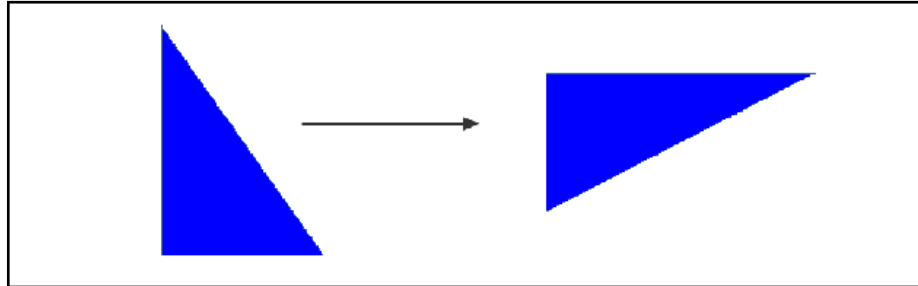


Fig.8-100: Rotating objects

This tool can only be used with a single selected object or a [group on page 235](#) object. This tool cannot be used with multiple objects selected.

"Mirroring Vertically"

Click on the tool , **Mirror vertically**, to invert the selected object vertically. The object rotates around an imaginary vertical center line until it is a mirror image of the original object:

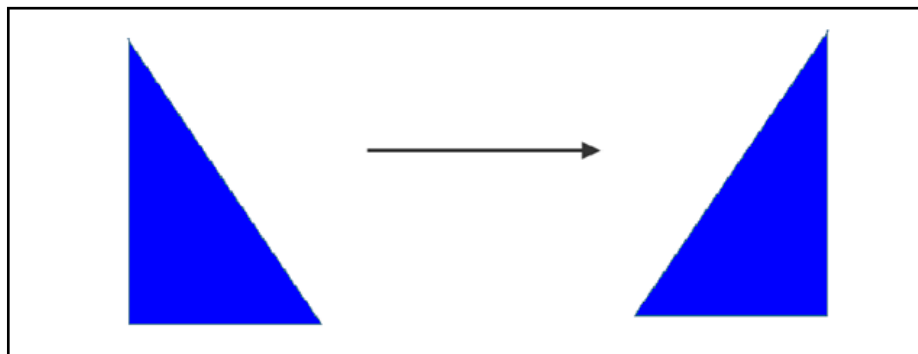


Fig.8-101: Mirroring objects vertically

This tool can only be used with a single selected object or a [group on page 235](#) object. This tool cannot be used with multiple objects selected.

"Mirroring Horizontally"

Click on the tool , **Mirror horizontally**, to invert the selected object horizontally. The object rotates around an imaginary horizontal center line until it is a mirror image of the original object.

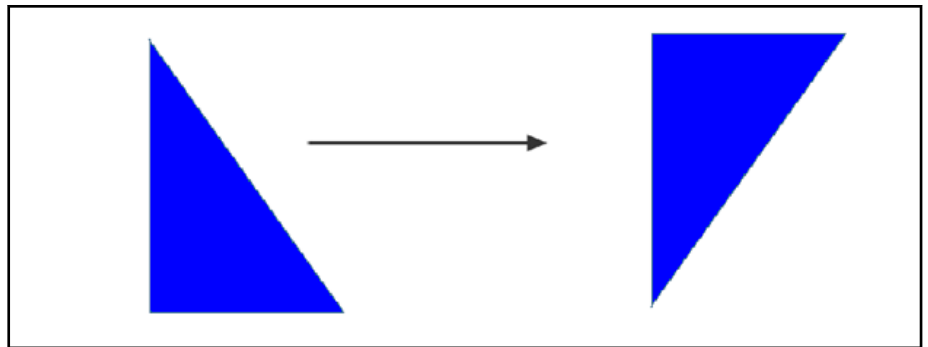


Fig. 8-102: *Mirroring objects horizontally*

This tool can only be used with a single selected object or a [group on page 235](#) object. This tool cannot be used with multiple objects selected.

8.4.7 Adjusting the Position

Moving a screen object by selecting and dragging it can be imprecise, particularly if "Snap to grid" is activated. For this reason, positioning tools can be used to exactly adjust (readjust) the position of objects.

1. Click on a screen object to select it
2. Keep the <SHIFT> key pressed and use the arrow keys to adjust (readjust) the horizontal and vertical position of the object. The step width is derived from the grid width of the grid lines

Also refer to:

- [Status Bar on page 87](#)
- [Grid Settings on page 167](#)

8.4.8 "Same Width" / "Same Height" for Selected Objects

Use the following functions in the "WinStudio Alignment Functions" toolbar to adjust the size of screen objects:

Same width for selected objects

Click on , **Same width for selected objects**, to increase/decrease the width of the object selected first to the width of the secondly selected object.

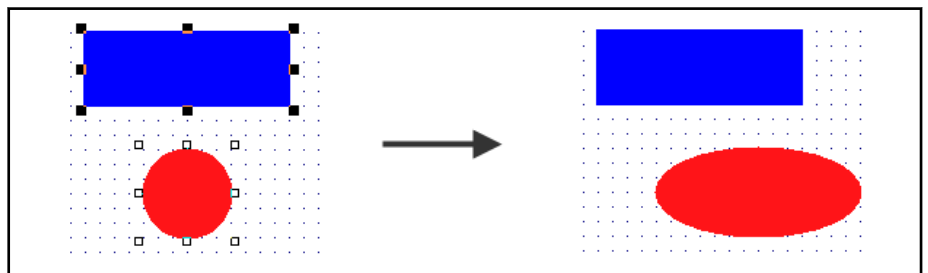


Fig. 8-103: *Elongating objects to the same width*

Same height for selected objects

Click on the tool , **Same height for selected objects**, to increase/decrease the height of the object selected first to the height of the secondly selected object.

Screens and Graphics

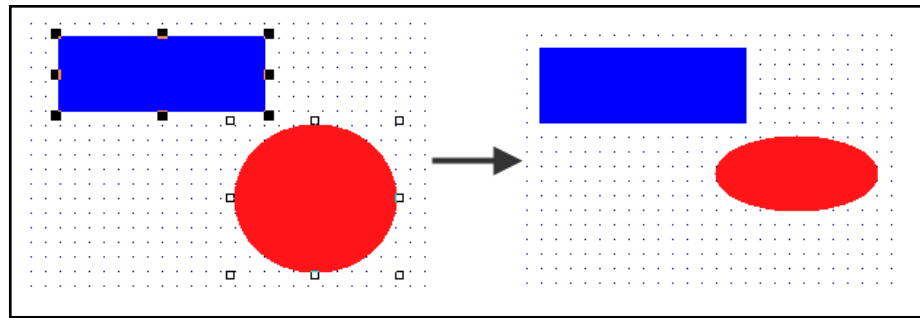


Fig.8-104: Elongating objects to the same width



By means of **Adjust width** and **Adjust height**, an ellipse can be turned into a circle or a rectangle into a square. However, only one object may be selected.

The mouse and arrow keys can also be used to resize objects. When selecting an object (or group of objects) with the cursor, handles are displayed at each corner and at centers of each edge. These handles can be used as follows:

- **To enlarge an object**, drag a handle into the respective direction. Dragging a side handle resizes the object in one direction only (height only or width only). Dragging a corner handle resizes the entire object (height and width). When dragging a corner handle, the proportion of sides remains unchanged by default. To freely resize the object, hold down the SHIFT key while dragging the handle
- **To adjust the size of an [open polygon on page 168](#) or [closed polygon on page 169](#)**, select and [group on page 235](#) the polygon (Group context menu). The size of the polygon can then be adjusted similar to any object



If the function to adjust the size is used for grouped objects (e.g. linked symbols, group of screen elements), all individual elements in the group are adjusted in the same direction and in the same proportion.

Also refer to:

- [Status Bar on page 87](#)

8.4.9 Fill Color

Click on the tool , **Fill Color**, on the "WinStudio Object Properties" toolbar to specify a default fill color for the following objects:

- [Closed Polygon Object on page 169](#)
- [Ellipse Object on page 172](#)
- [Rounded Rectangle Object on page 171](#)
- [Rectangle Object on page 170](#)



To save development time, several objects can be selected (of any type specified in the preceding list) to use **Fill Color** to specify a default fill color for all of them at once.

Also refer to:

- **Edit the color of a component:** [Color Interface on page 241](#)

8.4.10 Line Color

Click on the tool , **Line Color**, on the "WinStudio Object Properties" tool-bar, to specify a line color for selected objects or to set a default color for new objects, including the following:

- [Line Object on page 167](#)
- [Open Polygon Object on page 168](#)
- [Closed Polygon Object on page 169](#)
- [Rectangle Object on page 170](#)
- [Rounded Rectangle Object on page 171](#)
- [Ellipse Object on page 172](#)

Click the **Line Color** tool, to display the "Line Selection" dialog. Use this dialog to specify line styles and colors for the selected objects.



To save development time, several of the preceding objects can be selected simultaneously to use the **Line Color** tool to specify a line color for all of the objects at once.

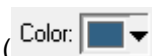
Also refer to:

- **Edit the color of a component:** [Color Interface on page 241](#)

8.4.11 Color Interface

The color interface can be used to edit the color of a component.

Color picker

1. Click on the symbol () in the "Properties" menu item of a graphic object
2. A popup box is displayed to select one of ten default colors or sixty coordinated design colors

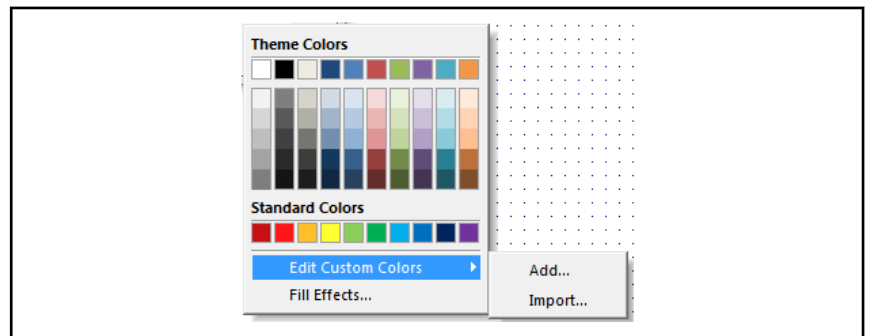


Fig. 8-105: Pop-up box with main colors

The selected color is applied to the edited component

The box enables transition to

- Editing user-defined colors: ["Adding standard and user-defined colors" on page 241](#)
- Editing user-defined colors: ["Import" on page 242](#)
- ["Fill effects" on page 242](#)

Adding standard and user-defined
colors

Screens and Graphics

1. Click on **Edit Custom Colors** to use a different color. The color can be selected on the **Standard** tab by clicking on the required color hexagon or by using the eyedropper (Select button)



The eyedropper enables applying any color from the IndraWorks environment.

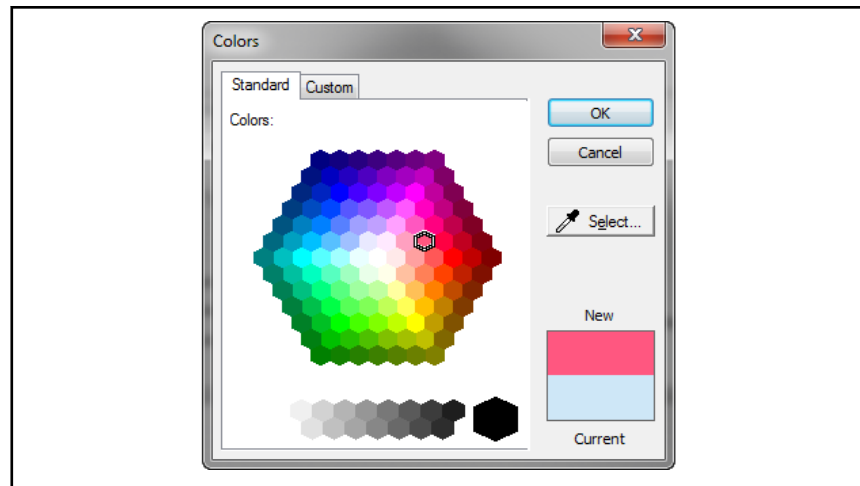


Fig.8-106: Pop-up box with standard colors

2. Click on the **Custom** tab.

The custom color can be defined by specifying the HSL (hue, saturation, lightness) color code or the RGB (red, green, blue) color code (also refer to ["RGB code table" on page 243](#))

As an alternative, practically any color can be selected by clicking with the mouse or using the eyedropper

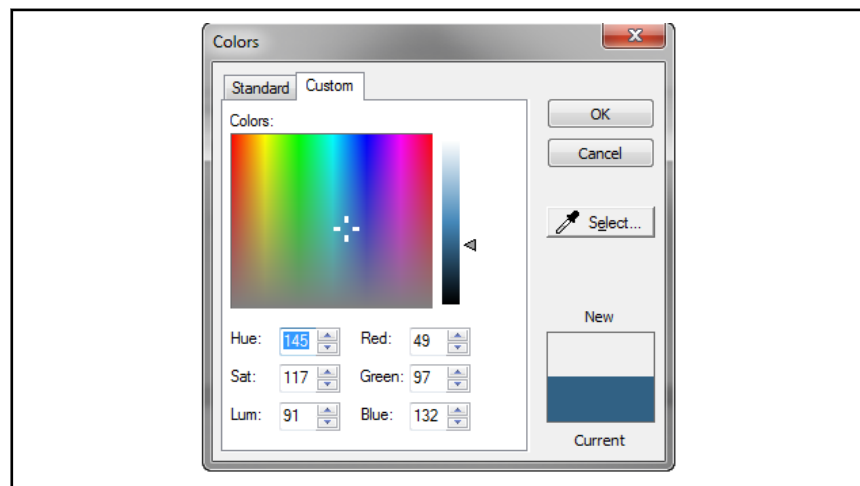


Fig.8-107: Pop-up box to create a custom color

3. Click on **OK** to apply the selected color to the component that is being edited

Import This option enables import of a "user-defined color" text file.

- Fill effects**
1. Depending on the component that is edited, the **Fill Effects** option is available on the pop-up interface. Click this option to apply gradient colors with different styles and variants

Screens and Graphics

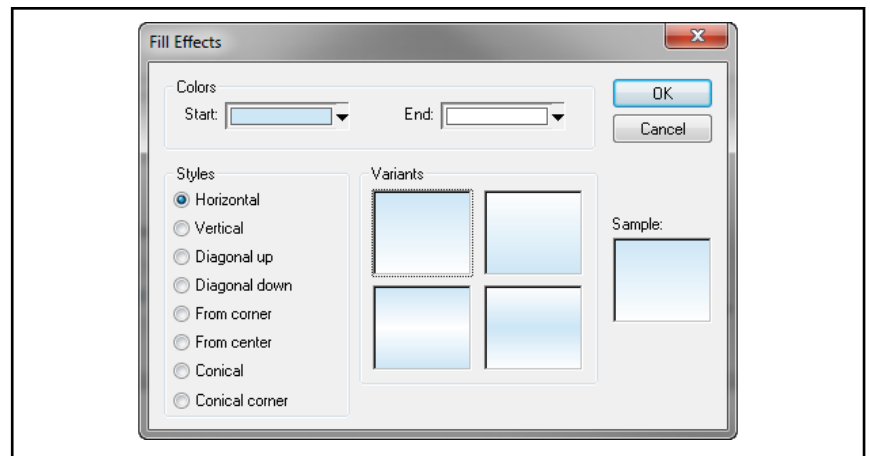


Fig. 8-108: Fill Effects dialog

2. Select two colors in the **Start** and **End** fields, select the "Style" and click on the desired "Variant". Finally, click **OK** to apply the fill effect to the component which is being edited

For projects on Windows CE target systems, the **Fill Effects** interface is available only for **Rectangle** objects.

Although **fill effects** are a useful tool for enhancing the look and effect of screens, the operating system takes longer to fill objects with fill effects than with plain colors. Particularly when creating Win CE user interfaces, this additional resource requirement is to be taken into consideration.

RGB code table

By means of the [color animation on page 227](#), the color of objects can be modified during runtime. When this animation is configured with type = "By Color", the color to be displayed in the object during runtime can be set by the color code. The following table provides the code values as well as the RGB values for the most commonly used colors:

Name	RGB code			Code value
	R (Red)	G (Green)	B (Blue)	
Black	0	0	0	0
Dark red	128	0	0	128
Red	255	0	0	255
Pink	255	0	255	16711935
Rose	255	153	204	13408767
Brown	153	51	0	13209
Orange	255	102	0	26367
Light orange	255	153	0	39423
Gold	255	204	0	52479
Light brown	255	204	153	10079487
Olive green	51	51	0	13107
Dark yellow	128	128	0	32896
Lime	153	204	0	52377
Yellow	255	255	0	65535

Screens and Graphics

Name	RGB code			Code value
	R (Red)	G (Green)	B (Blue)	
Light yellow	255	255	153	10092543
Dark green	0	51	0	13056
Green	0	128	0	32768
Sea green	51	153	102	6723891
Bright green	0	255	0	65280
Light green	204	255	204	13434828
Dark teal	0	51	102	7877376
Teal	0	128	128	8421376
Aqua	51	204	204	13421619
Turquoise	0	255	255	16776960
Light turquoise	204	255	255	16777164
Dark blue	0	0	128	8388608
Blue	0	0	255	16711680
Light blue	51	102	255	16737843
Sky blue	0	204	255	16737843
Pale blue	153	204	255	16764057
Indigo	51	51	153	10040115
Blue-gray	102	102	153	10053222
Violet	128	0	128	8388736
Plum	153	51	102	6697881
Lavender	204	153	255	16751052
Gray-80%	51	51	51	3355443
Gray-50%	128	128	128	8421504
Gray-40%	150	150	150	9868950
Gray-25%	192	192	192	12632256
White	255	255	255	16777215

The functions [RGB Color on page 601](#) and [RGB Component on page 602](#) can be used to adjust color codes during runtime.

The number of colors available when developing the project depends on the color settings configured on the operating system of the development station.

The number of colors available when running the project depends on the color settings configured on the operating system of the runtime station.

8.4.12 Fonts

Click on the tool  **Fonts**, on the "WinStudio Object Properties" toolbar to specify the font and color for selected [text objects on page 175](#) or to specify a default font and color for new text objects.



To save development time, several text objects can be selected simultaneously to apply the **Fonts** tool to specify font and color settings for all of the objects at once. (However, this function cannot be used for [grouped on page 235](#) text objects.)

"WinStudio" supports any UNICODE font available on the operating system. For this reason, it is possible to configure user interfaces for languages that do not use standard western characters, such as Japanese, Chinese, Arabic, Cyrillic, etc.

The font style of several objects can be changed simultaneously by selecting them all (press Shift while selecting the objects) and using the **Fonts** tool in the WinStudio toolbar.

The font of individual screen objects can also be adjusted by using the fonts button in the "Object Properties" window:

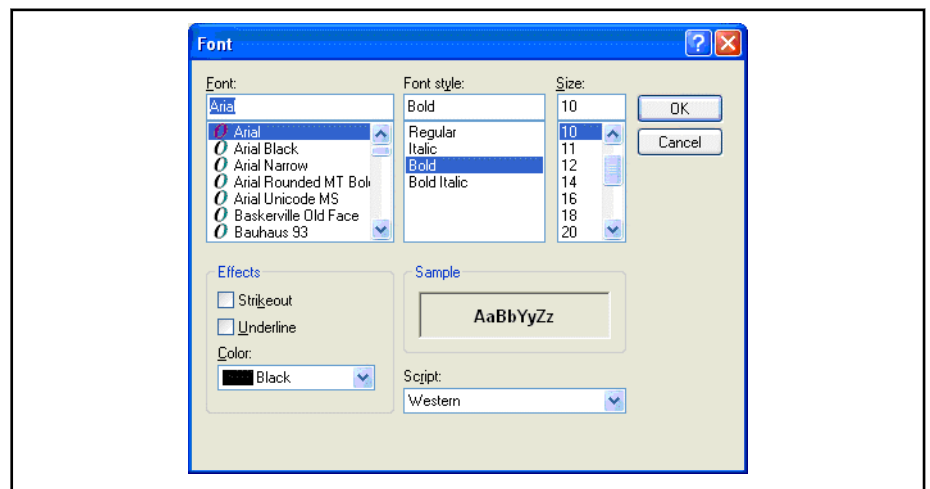


Fig. 8-109: Standard Fonts dialog

Font type, style, size, effects and script can be set.

The icon displayed to the left of the font name indicates the font format (technology)

Symbol	Technology	Information	
	TrueType	Outline font format	TrueType and OpenType fonts are outline fonts that consist of line and curve commands. OpenType is an extension of TrueType. Both can be scaled and rotated. Both can be displayed in all sizes and on all output devices supported by Windows
	OpenType		Windows provides a selection of OpenType fonts, including Arial, Courier New, Lucida Console, Times New Roman, Symbol, and Wingdings The font Type1 by Adobe Systems, Inc. is an outline font especially designed for PostScript printers. The "Outlines" can be scaled and rotated. With OpenType technology, Windows fully supports Type1 fonts
N/A	Vector		Vector fonts are supported as a number of programs still use this font format Vector fonts are rendered from a mathematical model. They are used primarily with plotters. Windows supports three vector fonts: Modern, Roman and Script
N/A	Grid		Raster fonts are supported as a number of programs still use this font format Raster fonts are stored in files as bitmap images and are composed of a series of dots both on the screen and on paper

Screens and Graphics

 CAUTION

It is strongly recommended to only use TrueType or OpenType fonts.

Fonts designed with other technologies (e.g. Courier) cannot be scaled properly and can cause issues during runtime.



When designing screens, the fonts available in the operating system of the development station are provided.

The fonts on the runtime station, however, may look different (e.g. different size in pixels), even if all settings are the same on both stations.

These situation occurs more frequently if projects are executed on CE operating systems where font sizes are not always displayed identically. For this reason, it is crucial to test the created screens during project development on the target platform.

9 Alarms, Events and Trends

9.1 Alarms, Events and Trends, General Information

The Alarm and Trend tasks are used to log historical data, and the Alarm/Event and Trend objects are used to display historical data on a project screen.

These two functions are usually used in combination, however, they do not need to be.

Project data may be logged without being displayed at runtime and the data displayed during runtime may be taken from outside the project.

9.2 Alarm Worksheet

9.2.1 Alarm Worksheet, General Information

The Alarms folder enables the configuration of alarm groups and tags related to each group. The Alarm worksheet defines the alarm messages generated by the application. The primary purpose of an alarm is to inform the operator of any problems or abnormal condition during the process so he can take corrective action(s).

The Alarm worksheet is executed by the Background Task module (refer to <Execution Tasks>). It manages the state of all alarms and saves the alarm messages to the history if configured to do so. The Alarm worksheet itself, however, does not display alarm messages at runtime. To display alarms, the [Screen object Alarm/Event Control on page 262](#) in the "WinStudio Screen Tools" tool window has to be inserted into a WinStudio screen and configured.

To create a new worksheet, open the context menu of the "Tasks" folder and select **Add ► Alarm**.

To edit an existing alarm worksheet, double-click the element in the Project Explorer.

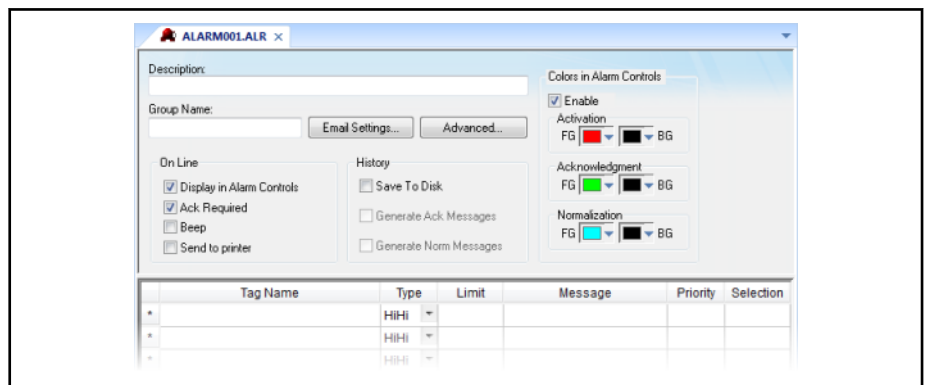


Fig. 9-1: Alarm worksheet

Multiple alarm groups (worksheets) can be created and each group can be configured with independent settings, such as message colors, history log enabled/disabled, etc.

Each alarm worksheet is composed of two areas:

- **Header:** Settings applied to all tags and alarms configured in the same alarm group. These settings enable configuration of the formatting of the messages and actions that have to be triggered based on alarm events

Alarms, Events and Trends

(e.g., print alarms, send alarms by email, etc.). For further information, refer to [header settings on page 248](#).

- **Body:** Selecting tags and defining conditions on which alarm messages are created. For further information, refer to [body settings on page 253](#).



- Alarm groups can be configured to send notifications by Email automatically, based on alarm events. For further information, refer to [email settings on page 250](#).
- The alarm properties associated with each tag (configured in the body of the alarm group) can also be edited on the [Tag Properties on page 140](#) dialog (**Properties** in the "Cross references" tool window). However, before associating a tag with an alarm group, it is necessary to create the alarm group and to configure the settings on its header. These will be applied to all tags associated with the group.
- As of "WinStudio" v6.1+SP2, the Alarm task has been modified to avoid automatically acknowledging alarms by another alarm. For example, the Hi (Lo) alarm is not to be automatically acknowledged when the HiHi (LoLo) alarm becomes active. To enable this behavior, set the following key in the project file (.APP):

[Alarm]

UseLegacyPriorityAck=1

NOTICE

The settings configured in the body of each alarm worksheet are stored in the tags database archive(s). Therefore, changes to the tags database may affect the content of the alarm worksheets (body).

Please note that each tag/type cannot be available in more than one alarm group simultaneously as the alarm group is a property associated to each tag/alarm type

(e.g., Tag: Level; Alarm Type: Hi; Alarm Group: 2).

9.2.2 Alarm Worksheet Header

Alarm Worksheet Header, General Information

Fig. 9-2: Header settings on an alarm worksheet

The following table describes the Header settings on an alarm worksheet (refer to [Alarm Worksheet on page 247](#)):

Alarms, Events and Trends

Field	Information	Syntax
Description	Description of the alarm group. It is displayed on the workspace. This field is used for documentation only.	Text (up to 80 characters)
Group Name	Name of the alarm group. At runtime, the operator can filter alarms based on the group name by the built-in Filter dialog of the Alarm/Event Control object.	Text (up to 32 characters)
Email settings	Opens the Email Settings dialog on page 250 , to configure the settings for emails sent automatically based on alarm conditions.	Button
Advanced	Opens the Advanced Settings dialog on page 252 , to configure the settings for emails sent automatically based on alarm conditions.	Button
Online > Display in Alarm Controls	If this option is enabled, the alarms are available to be displayed on the Alarm/Event Control object on page 262 .	Checkbox
Online > Ack Required	If this option is enabled, the alarms require acknowledgment. In this case, the alarms are displayed on the Alarm/Event Control object on page 262 (Online mode) until they are acknowledged AND normalized.	Checkbox
Online > Beep	If this option is enabled, the computer keeps beeping, if there are currently active alarms to be acknowledged.	Checkbox
Online > Send to Printer	If this option is enabled, the alarm messages are sent to the printer as soon as the alarm event occurs. This option requires a matrix printer (instead of an inkjet or laser printer) in order to print the message(s) and it has to be fed by just one line, otherwise, each alarm is printed on a different sheet of paper. The alarms are printed on the default printer. If alarms are to be sent to a printer different from the default printer, the printer path/name can be specified by editing the following parameter in the project_name.APP file: [AlarmLog] Device=PrinterPath/PrinterName	Checkbox
History > Save to Disk	If this option is enabled, the alarm messages are stored in the History log on page 255 when they become active.	Checkbox
History > Generate Ack Messages	If this option is enabled, the alarm messages are stored in the History log on page 255 when they are acknowledged.	Checkbox
History > Generate Norm Messages	If this option is enabled, the alarm messages are stored in the History log on page 255 when they are normalized.	Checkbox

Alarms, Events and Trends

Field	Information	Syntax
Colors in Alarm Controls > Enable	If this option is enabled, the alarms configured in this group are be displayed with the colors assigned to each alarm state (Activation , Acknowledgement or Normalization), according to the colors configured in the alarm group.	Colors on page 241
Colors in Alarm Controls > FG and BG	Text foreground colors (FG) and background colors (BG) for the alarms displayed can be configured on the Alarms/Events Control object on page 262 . Each alarm state can be displayed with a different color scheme: Activation: Alarm active and not acknowledged Acknowledge: Alarm active and acknowledged Normalization: Alarm no longer active and not acknowledged.	Colors on page 241

Email Settings for Alarm Worksheet

"WinStudio" offers the function to send emails automatically when alarm events occur. Emails are sent using the standard SMTP (Simple Message Transfer Protocol). Therefore, a valid email account with an SMTP and POP3 server is required. It is not necessary to install any additional software, such as Microsoft Outlook.



Before emails can be sent, it is necessary to execute the [CnfE-mail on page 658](#) function at least once successfully (from the internal WinStudio language). This function sets the email account parameters used when sending emails from the project (e.g., SMTP server, user name, password, etc.).

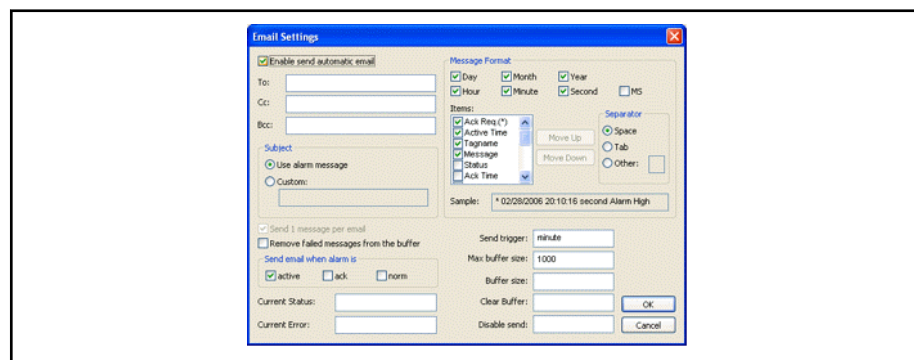


Fig.9-3: Alarms Worksheet – Email Settings

Field	Information	Syntax
Enable send automatic email	Name of the tag associated with the alarm.	Checkbox
To, Cc, Bcc	Target addresses for the emails. Multiple email addresses can be configured in each box (To, Cc and/or Bcc) by separating the addresses with a semi-colon (",").	Text and/or {Tag} (up to 1024 chars)
Subject	When selecting "Use alarm message", the alarm message itself is used as the subject of the email to be sent. When selecting "Custom", a custom text can be configured as Subject when sending the alarm.	Radio button / Text (up to 1024 chars)

Alarms, Events and Trends

Field	Information	Syntax
Send 1 message per email	If this option is enabled, each alarm is sent in an individual email and all emails are sent when the Send trigger is triggered. Otherwise, all alarm messages are buffered and sent in only one email when the Send trigger is triggered. This option cannot be disabled (unchecked) when "Use alarm message" is configured under Subject.	Checkbox
Remove failed messages from the buffer	If this option is enabled, emails are removed from the buffer after attempting to send them even if there was an error (failure) and the email was not sent. Otherwise, the messages are kept in the buffer until they are sent successfully or when the buffer reaches its maximum size.	Checkbox
Send email when alarm is	Enables configuration of which alarm events are to generate emails: Please note that each event can be enabled/disabled individually. • Active: When the alarm becomes active. • Ack: When the alarm is acknowledged. • Norm: When the alarm is normalized.	Checkbox
Current status	The tag configured in this field, if any, is updated with the current status of the current or last email that the project attempted to send: • "-2": Incorrect version of the INDMail.DLL library. • "-1": The INDMail.DLL library is corrupted. • " 0": The function SendEmailExt on page 662 is not being executed. • " 1": Sending email(s) • " 2": Last email was sent successfully. • " 3": There was an error sending the last email.	Tag
Current error	The tag configured in this field, if any, is updated with the error message describing the result of the last email that the project attempted to send. Therefore, when configuring a tag in this field, this tag has to be a String type.	Tag
Message format	This interface enables configuration of the actual format of the message sent by email, based on the alarm event(s): When configuring these settings, the Sample field displays an example of the format of the message according to the settings being configured. • Day, Month, Year, Hour, Minute, Second, MS: The enabled options define the timestamp for the alarm messages. MS stands for milliseconds. • Items: The enabled options define the email message for each alarm. The order of the items can be configured by using the Move Up and Move Down buttons. • Separator: Enables choosing the separator used between the items checked in this interface.	Checkbox and radio button

Alarms, Events and Trends

Field	Information	Syntax
Send trigger	When alarm events are generated, they are kept in an internal buffer (memory). If the tag configured in this field changes its value, the email(s) on the internal buffer are sent to the addresses configured in the To , Cc and Bcc fields. After being successfully sent, the emails are removed from the internal buffer.	Tag
Max buffer size	Maximum number of alarm messages (events) that can be stored in the internal buffer simultaneously. If this limit is reached, the buffer follows a FIFO (First In, First Out) principle, deleting older messages as soon as newer messages are generated. This way, it is guaranteed that the buffer does not exceed the limit configured in this field.	Tag or Number
Buffer size	The tag configured in this field, if any, is updated with the number of messages (events) currently stored in the internal buffer.	Tag
Clear buffer	If the tag configured in this field changes its value, all messages (events) currently stored in the buffer are deleted. These messages are never sent.	Tag
Disable send	If the value of the tag configured in this field is TRUE, the Email function is temporarily disabled. Alarm events generated while the Email function is disabled are not stored in the internal buffer. Also, emails will NOT be sent in this condition, even if the tag configured in the field Send trigger changes its value.	Tag

Advanced Settings for Alarm Worksheet

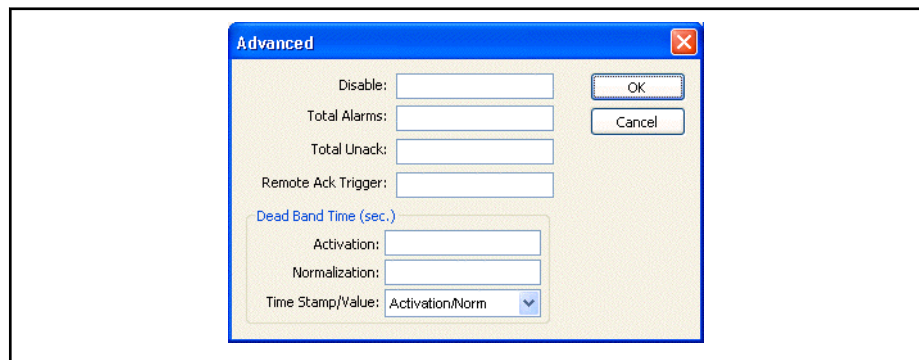


Fig.9-4: Alarm Worksheet, Advanced settings dialog

The following table describes the Advanced settings on an [Alarm worksheet](#) on page 247:

Field	Information	Syntax
Disable	When the value of the tag configured here is TRUE, all alarms configured in this group are temporarily disabled. This option is useful to disable alarms under special conditions (e.g., during maintenance).	Tag
Total alarms	The tag configured in this field, if any, is updated with the number of alarms from this group, which are currently active.	Tag

Alarms, Events and Trends

Field	Information	Syntax
Total unack	The tag configured in this field, if any, is updated with the number of alarms from this group, which are currently active AND have not been acknowledged yet.	Tag
Remote ack trigger	If the tag configured in this field changes its value, all active alarms from this group are acknowledged. This option can be used to acknowledge alarms regardless of any action from the operator.	Tag
Dead band time > activation	Each alarm has to remain continuously in its alarm condition for the period of time specified in this field before becoming active. This option is useful to avoid generating alarms on intermittent conditions (e.g., noise). If this field is left in blank, the alarm becomes active as soon as its condition is true.	Tag or Number
Dead band time > normalization	Each alarm has to remain continuously out of its alarm condition for the period of time specified in this field before becoming normalized. This option is useful to avoid normalizing alarms on intermittent conditions (e.g., noise). If this field is left in blank, the alarm become normalized as soon as its condition is no longer true.	Tag or Number
Dead band time > time stamp/value	Each alarm maintains a time stamp of the last significant activity, along with the value of the tag at that time. The type of activity that updates the time stamp: • Activation/Norm (default): The time when the dead band ended, i.e., when the alarm becomes activated or normalized. • Last Tag Change: The time when the value of the tag last changed during the dead band. • Start Condition: The time when the dead band started.	Combo box

9.2.3 Alarm Worksheet Body

	Tag Name	Type	Limit	Message	Priority	Selection
*		HiHi				
*		HiHi				
*		HiHi				

Fig. 9-5: Alarm worksheet body

The following table describes the Body settings on an [Alarm worksheet on page 247](#):

Alarms, Events and Trends

Field	Information	Syntax
Tag name	Name of the tag associated with the alarm.	Tag
Type	Type of the alarm: When using the types Rate, DevP and DevM, it is necessary to configure additional settings in the Tag Properties dialog (Tag Properties in the "Cross references" tool window). • HiHi: Activates the alarm if the tag value is equal or higher than the limit. • Hi: Activates the alarm if the tag value is equal or higher than the limit. (For Boolean tags, if the value is 1.) • Lo: Activates the alarm if the tag value is equal or lower than the limit. (For Boolean tags, if the value is 0.) • LoLo: Activates the alarm if the tag value is equal or lower than the limit. • Rate: Activates the alarm if the tag value varies faster than the rate specified to the alarm. (For Boolean tags, if the value changes.) • Dev+: Activates the alarm if the tag value is equal or higher than the Set Point tag plus the limit. • Dev-: Activates the alarm if the tag value is equal or lower than the Set Point tag minus the limit.	Combo box
Limit	Limit associated with each alarm. The limits can be modified dynamically during runtime by the tag fields HiHiLimit, HiLimit, LoLimit, LoLoLimit, Rate, DevP and DevM (e.g., TagLevel->HiLimit).	Count
Message	Message associated with the alarm. The message can be displayed on the Alarm/Event Control object on page 262 and/or stored in the Alarm History on page 255 and/or sent by Email on page 250 , depending on the settings configured in the Header on page 248 of the alarm group.	Text and/or {Tag} (up to 256 chars)
Priority	Priority number associated with the alarm. When displaying alarms on the Alarm/Event Control object on page 262 , the operator can filter and/or sort the alarms by priority.	Number (from 0 to 255)
Selection	Alias associated with the alarm (e.g., AreaA, AreaB, etc). When displaying alarms on the Alarm/Event Control object on page 262 , the operator can filter and/or sort the alarms by their selection value.	Text (up to 7 characters)

9.2.4 Saving the Alarm History to an External Database

By default, the alarm history and event log are saved to proprietary-format text files in the Alarms folder of the project. However, the project settings can be modified to save them to an external SQL database instead.

1. Open the **Options** tab in the "Project Settings" dialog.

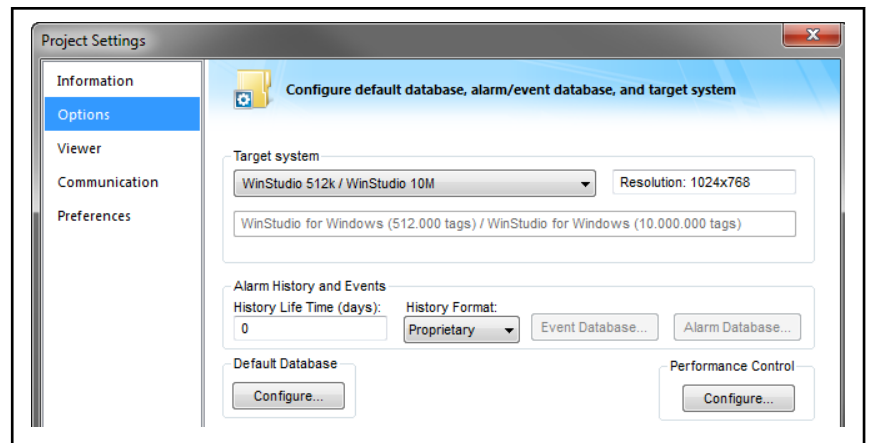


Fig.9-6: Project Settings: Options

2. In the **Alarm History and Events** pane, in the **History Life Time (days)** input field, enter the number of days of history that are to be saved. As the history exceeds the specified number of days, it is automatically deleted in a first-in, first-out manner. If no number is specified - that is, if it is left blank or set to 0, - the history is never deleted. There is no limit on how much history can be saved, however, the more is saved, the more disk space it takes.
3. From the **History Format** list, select **Database**.
4. To configure a single, default database to be used for both the alarm history and the event log (as well as all other runtime tasks), click **Configure** in the **Default Database** pane. The "Default Database Configuration" dialog is displayed. Configure the database connection using this dialog. For further information, refer to [Configuring a Default Database for All Task Histories on page 442](#).
5. To configure a separate database, either for the event log or the alarm history, click **Event Database** or **Alarm Database**, respectively. In either case, a "Default Database Configuration" dialog is displayed. Configure the database connection using this dialog. For more information, refer to [Default Database Configuration on page 438](#).
6. Click on **OK**.

9.2.5 Format of the Alarm History

The location and format of the alarm history depends on whether the History Format is set to Proprietary or Database. This section describes both.

If the History Format is Proprietary (default), the alarm history is saved as a series of text files in the project Alarm folder, with one file per calendar day. The name of each file is "AL"yymmdd".ALH", where:

- **yy** is the last two digits of the year in which the alarm history file was generated;
- **mm** is the month in which the alarm history file was generated; and
- **dd** is the day of the month on which the alarm history file was generated.

Therefore, the alarm history file for 07 May 2003 can be found at

[...]Alarm\AL030507.ALH.



To change the location where these files are saved, use the function [SetAppAlarmPath on page 628](#).

Alarms, Events and Trends

Within a *.ALH file for a specific day, each alarm is saved as a new line using the pipe character ("|") to delimit the fields as illustrated below:

*.ALH file

```
P1|P2|P3|P4|P5|P6|P7|P8|P9|P10|P11|P12|P13|P14|P15|P16|P17|P18|P19|P20|P21|P22|P23
P1|P2|P3|P4|P5|P6|P7|P8|P9|P10|P11|P12|P13|P14|P15|P16|P17|P18|P19|P20|P21|P22|P23
.
P1|P2|P3|P4|P5|P6|P7|P8|P9|P10|P11|P12|P13|P14|P15|P16|P17|P18|P19|P20|P21|P22|P23
```

If the History Format is set to Database, the alarm history is saved as a table in the database that is configured as alarm database. For further information, refer to [Saving the alarm history to an external database on page 254](#).

The fields/columns of the alarm history are described below:

Proprietary	Database		Description	File vers.
Field number	Column name	Data type		
P1	—	—	File version (current = 003)	001
P2	Al_Start_Time	Time stamp	Start date (MM/DD/YYYY)	001
P3			Start time (HH:MM:SS)	001
P4	Al_Tag	String	Tag name	001
P5	Al_Message	String	Alarm message	001
P6	Al_Ack	Boolean	Ack, where: 0: Alarm was acknowledged or does not require acknowledgment 1: Alarm was not acknowledged	001
P7	Al_Active	Boolean	Active, where: 0: Alarm is not active 1: Alarm is active	001
P8	Al_Tag_Value	Real	Tag value when the alarm occurred	001
P9	Al_Group	Integer	Alarm group number	001
P10	Al_Priority	Integer	Priority number	001
P11	Al_Selection	String	Selection	001
P12	Al_Type	Integer	Type, where: 1 is HiHi 2 is Hi(On) 4 is Lo(Off) 8 is LoLo 16 is Rate(Change) 32 is Deviation+ 64 is Deviation-	001
P13	Al_Ack_Req	Boolean	Ack required, where: 0: Alarm requires acknowledgement 1: Alarm does not require acknowledgement	001

Alarms, Events and Trends

Proprietary	Database		Description	File vers.
Field number	Column name	Data type		
P14	AI_Norm_Time	Time stamp	Normalization date (MM/DD/YYYY)	001
P15			Normalization time (HH:MM:SS)	001
P16	AI_Ack_Time	Time stamp	Ack date (MM/DD/YYYY)	001
P17			Ack time (HH:MM:SS)	001
P18	AI_User	String	User name	002
P19	AI_User_Comment	String	Comment	002
P20	AI_User_Full	String	User full name	003
P21	AI_Station	String	Station	003
P22	AI_Prev_Tag_Value	Real	Previous value	003
P23	Bias	Integer	Time zone bias	003
—	AI_Start_Time_ms	Integer	Number of milliseconds for the start time time stamp. This field is used if the database does not support ms in a time stamp field.	003
—	AI_Norm_Time_ms	Integer	Number of milliseconds for the norm time time stamp. This field is used if the database does not support ms in a time stamp field.	003
—	AI_Ack_Time_ms	Integer	Number of milliseconds for the ack time time stamp. This field is used if the database does not support ms in a time stamp field.	003
—	AI_Deleted	Boolean	Deleted, where: 0: Alarm message was not deleted by the user (not visible). 1: Alarm message was deleted by the user (visible).	003
—	Last_Update	Time stamp	Time stamp of the last update for this alarm.	003
—	Last_Update_ms	Integer	Number of milliseconds for the "Last_Update" time stamp. This field is used if the database does not support ms in a time stamp field.	003



The names of the columns in the database table can be individually adapted by manually editing the project file

(e.g., [...]projectname.APP) as follows:

[Alarm]

DefaultName=NewName

For example:

[Alarm]

Message=Alarm_Message

Ack=Acknowledgement

Alarms, Events and Trends

9.3 Events and Event Logs

9.3.1 Events and Event Logs, General Information

This section describes the logging and event-retrieval features of "WinStudio". Events can be tag changes, generating reports or recipes, opening and closing of screens, logging on/off at/from the security system, etc. "WinStudio" stores all these events in a log file that can be retrieved by the Alarm/Event Control object.

9.3.2 Enabling the Event Logger

Event logging is disabled by default to conserve runtime resources. To enable saving of events to the history file, use the Event Logger in the Project Explorer.

1. Open the context menu of the "WinStudio" folder and select the entry **Tools ► Event Logger**. The "Event Logger" dialog is displayed.

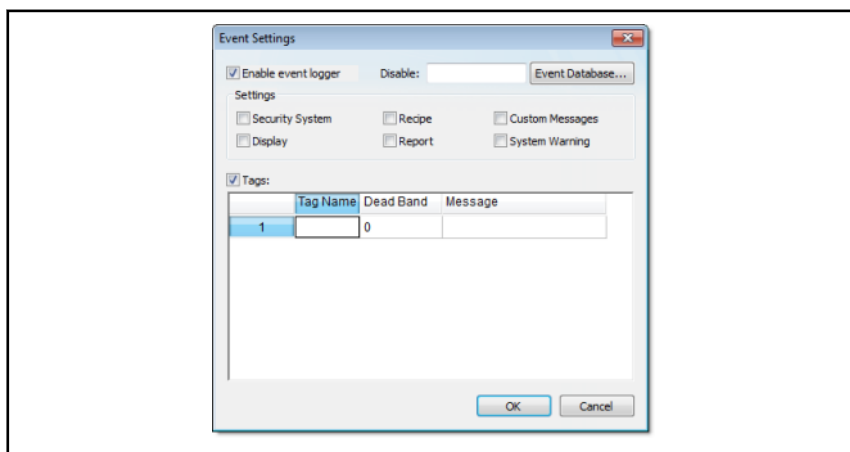


Fig.9-7: Event Settings dialog

2. Select **Enable event logger**.
3. In the **Disable** field, type the name of an application tag. If the value of the tag is TRUE (i.e., not zero) at runtime, event logging is suspended.
4. In the **Settings** pane, select the types of events that are to be logged to the history file.

Option	Description
Security system	Events generated by the project's security system, including: - User logged on/off - User created/removed by means of the functions CreateUser on page 536 or RemoveUser on page 543. - User blocked/unblocked by means of the functions BlockUser on page 535 or UnblockUser on page 545. - User blocked by the security system after several attempts to enter an invalid password - Password expired - Password changed - Invalid log on attempt
Display	Open Screen and Close Screen events.
Recipe	Recipes loaded, saved, initialized or deleted.

Alarms, Events and Trends

Option	Description
Report	Reports saved to disk or sent to printer.
Custom messages	Events created by means of the function SendEvent on page 686 .
System warning	Various runtime warnings and errors, including: Errors that occur when sending alarms by email Tag was blocked/unblocked Division by zero Connection/Disconnection of the remote security system

- To log changes in certain application variables, activate the **Tags** pane and specify the variables in the table.

The **Tags** option is useful for logging events that are not sufficiently important to be alarm conditions (for example, "Motor On", "Motor Off", etc.).

Column	Description
Tag name	The name of the application variable to be logged in the history file.
Dead band	A value to filter changes against, so that only changes greater than this value are logged (dead band). For example, if a dead band of 5 is defined for a tag value of 50 that changes to 52, the system does not register this change in the event log as the change is less than 5. However, if the tag value change is at least 5, the system logs the new value in the history file.
Message	A string (message) related to this tag change. Tags can be specified in messages using the {tagname} syntax.

The **Tags** option is useful for logging events that are not sufficiently important to be alarm conditions (for example, Motor On, Motor Off, etc.).

- Click on **OK**.

By default, the event log is saved as a series of text files in the project alarms folder. For further information, refer to [Format of the event log on page 260](#).

Alternatively, the event log can be saved to an external SQL database. For further information, refer to [Saving the event log to an external database on page 254](#).

9.3.3 Saving the Event Log to an External Database

By default, the alarm history and event log are saved to proprietary-format text files in the Alarms folder of the project. However, the project settings can be modified to save them to an external SQL database instead.

- Open the **Options** tab in the "Project Settings" dialog.

Alarms, Events and Trends

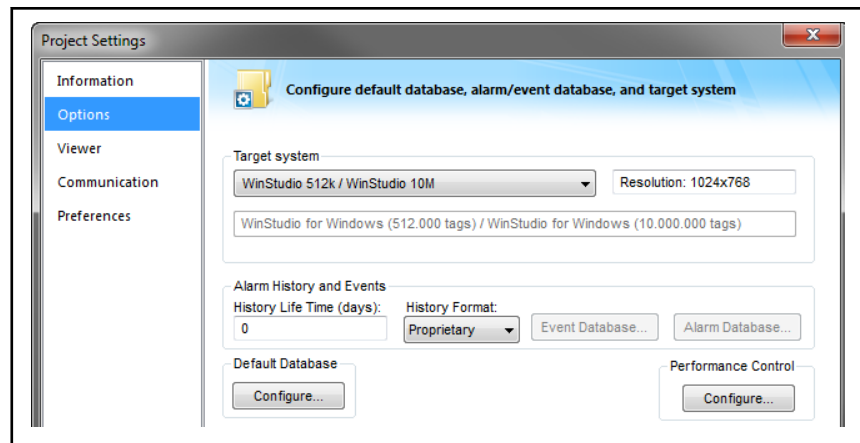


Fig.9-8: Project Settings: Options

2. In the **Alarm History and Events** pane, in the **History Life Time (days)** input field, enter the number of days of history that are to be saved. As the history exceeds the specified number of days, it is automatically deleted in a first-in, first-out manner. If no number is specified - that is, if it is left blank or set to 0, - the history is never deleted. There is no limit on how much history can be saved, however, the more is saved, the more disk space it takes.
3. From the **History Format** list, select **Database**.
4. To configure a single, default database to be used for both the alarm history and the event log (as well as all other runtime tasks), click **Configure** in the **Default Database** pane. The "Default Database Configuration" dialog is displayed. Configure the database connection using this dialog. For further information, refer to [Configuring a Default Database for All Task Histories on page 442](#).
5. To configure a separate database, either for the event log or the alarm history, click **Event Database** or **Alarm Database**, respectively. In either case, a "Default Database Configuration" dialog is displayed. Configure the database connection using this dialog. For more information, refer to [Default Database Configuration on page 438](#).
6. Click on **OK**.

9.3.4 Format of the Event Log

The location and format of the event log depend on whether the History Format is set to Proprietary or Database. This section describes both.

If the History Format is Proprietary (default), the event log is saved as a series of text files in the project Alarm folder, with one file per calendar day. The name of each file is "EV"ymmdd".EVT", where:

- **yy** is the last two digits of the year in which the event log file was generated;
- **mm** is the month in which the event log file was generated; and
- **dd** is the day of the month on which the event log file was generated.

Therefore, the alarm history file for 07 May 2003 can be found at [...]Alarm\EV030507.EVT.

To change the location where these files are saved, use the function [SetAppAlarmPath on page 628](#).

Within a *.EVT file for a specific day, each event is saved as a new line using the pipe character ("|") to delimit the fields, as illustrated below:

Alarms, Events and Trends

**.EVT*

P1|P2|P3|P4|P5|P6|P7|P8|P9|P10|P11|P12|P13
P1|P2|P3|P4|P5|P6|P7|P8|P9|P10|P11|P12|P13
.
.
P1|P2|P3|P4|P5|P6|P7|P8|P9|P10|P11|P12|P13

If the History Format is set to Database, the event log is saved as a table in the database that is configured as event database. For further information, refer to [Saving the event log to an external database on page 254](#).

The fields/columns of the event log are described below:

Proprietary	Database		Description
Field number	Column name	Data type	
P1	Version	Integer	This field is created only when the History format is Proprietary. Current version: 002
P2	Event_Type	Integer	1 Security System
			2 Display
			3 Recipe
			4 Report
			5 Custom message
			6 System warning
			7 Log tags
P3	Event_Time	Time stamp	Time stamp indicating when the event occurred. When the History Format is set to Proprietary, the project saves the event time in the following format: MM/DD/YYYY HH:MM:SS.MSS . If the History Format is set to Database, the project saves the event time in the default time stamp format of the database.
P4	Event_Info	String	Tag name.
P5	Value	Real	Tag value when the event occurred.
P6	Source	String	Name of the task that generated the event.
P7	User	String	User logged on when the event occurred.
P8	User_Full	String	Full name of the user logged on when the event occurred.
P9	Message	String	Event message.
P10	Station	String	Name of the station (computer) where the event occurred.
P11	Comment	String	Comment (optional) typed by the operator when the event occurred. This field is only available for Version >= 2.
P12	Previous_Value	Real	Tag value that occurred before the event. This field is only available for Version >= 2.
—	Deleted	Boolean	This field is created only if the History Format is set to Database. 0 (FALSE): Event message was not deleted. 1 (TRUE): Event message was deleted.

Alarms, Events and Trends

Proprietary	Database		Description
Field number	Column name	Data type	
P13	Bias	Integer	Difference (in minutes) between Event_Time and the GMT time. This field is only available for Version >= 2.
—	Last_Update	Time stamp	Time stamp indicating when the register was created/modified. This field is used to synchronize the databases when using the Secondary Database in addition to the Primary Database. This field is created only if the History format is set to Database.

The names of the columns in the database table can be individually adapted by manually editing the project file

(e.g. [...]My Documents\WinStudio v8.0 Projects\project_name\Web\Translation.trn)

as follows:

[EventLogger]

DefaultName=NewName

For example:

[EventLogger]

Event_Info=Information

Message=Event_Message

9.4 Alarm/Event Control Object

Use the **Alarm/Event Control** tool to add an Alarm or Event Control object to an project screen.

To create and configure an Alarm/Event Control object, proceed as follows:

1. In the "WinStudio Screen Tools" tool window, click on **Alarm object**.
2. Click in an open WinStudio screen and drag the mouse to create and adjust the shape of the object.
3. Double-click on the object to open the following "Object Properties" dialog.

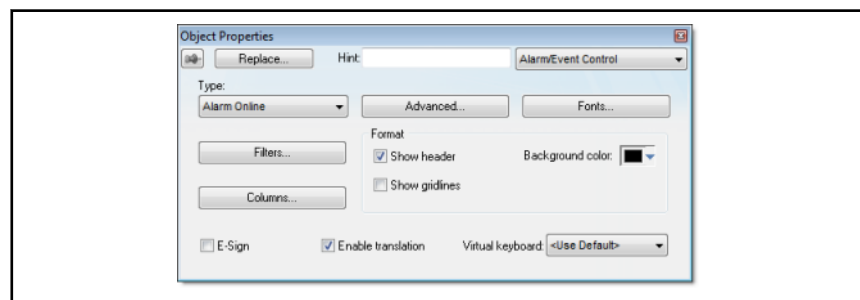


Fig.9-9: Object Properties: Alarm/Event Control

In this dialog, the following parameters can be specified:

- Select an alarm object mode in the "Type" pane:
 - **Alarm Online:** Displays only current alarm messages.
 - **Alarm History:** Displays only alarm messages from the Alarm History database.

Alarms, Events and Trends

- **Alarm History + Event:** Displays both alarm messages from the Alarm History database and logged events from the Event History database.
- **Event:** Displays only logged events from the Event History database.
- Click on (enable) the **Show gridlines** checkbox to display gridlines in the object.



Fig.9-10: Displaying gridlines

- Click on (enable) the **Show header** checkbox to display a header on the object.

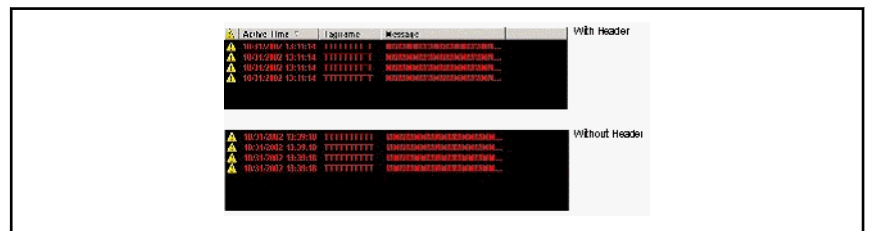


Fig.9-11: Displaying a header

- Use the **Background color** field to select a background color for the object. Click on the color box to open the color palette pop-up and simply click a color to select it.
- Click on (enable) the **Enable translation** checkbox to enable external translation of messages using the Translation tool. (Refer to [Translation tool on page 381](#) for further information.)
- **E-Sign** checkbox: If this option is enabled, a prompt is displayed to enter the electronic signature before executing the animation
- **Virtual keyboard:** Select the virtual keyboard type used for this object. Before configuring the virtual keyboard for this interface, the "Virtual keyboard" option has to be selected in the "Viewer" settings (**Viewer** tab in the WinStudio Properties dialog).

Fonts Click on the **Fonts** button to open a [standard fonts interface on page 244](#) to specify display properties for the message text.

Columns Click on the **Columns** button to open the "Columns" dialog to specify display properties for columns in the object.

Alarms, Events and Trends

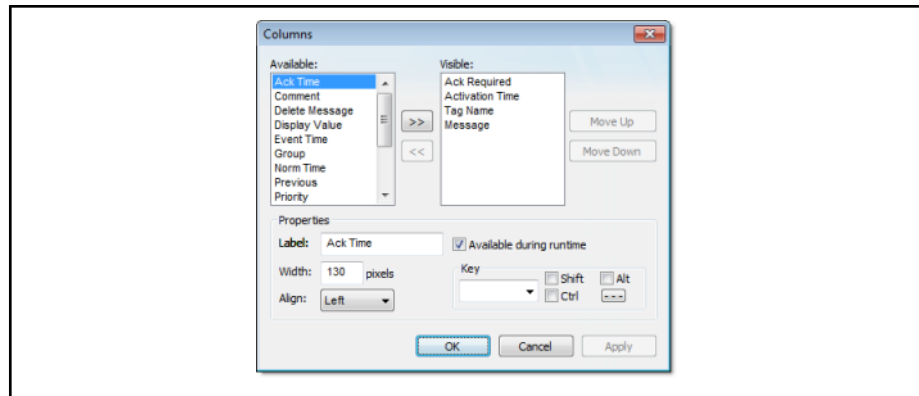


Fig.9-12: Columns dialog

- The "Available" list contains all of the column types available for this object. The "Visible" list contains all of the column types currently in use for this object. Click the » and « buttons to move selections between the two lists.



Alarm objects can be configured to display recently replaced values together with their new values. To do so, move both **Value** and **Previous** to the "Visible" list. Click on the **Move Up** or **Move Down** buttons to rearrange the order of columns in the "Visible" list.

- Use the **Label** and **Width** fields in the "Properties" pane to change the default column labels and widths at runtime.
- Use the **Align** combo box to specify alignment (**Left**, **Center**, or **Right**) for the alarm message text within a specified column.
- Click on (enable) the **Available during runtime** checkbox to allow the user to add selected columns to the Visible list at [runtime on page 262](#).
- Use the **Key** box to assign a shortcut to each column. This way, the information on the alarm control object can be sorted by any column, using keyboard keys instead of the mouse cursor.

To finish the settings, click **OK** to close the "Columns" dialog.



When acknowledging an alarm, the alarm control object sends a message to the [alarm task on page 247](#) with the following information: **Tag Name**, **Type**, **User** and **Station**. This is a solution to control acknowledged alarms from a Thin Client.

Filter To filter alarm messages at runtime, click the **Filters** button. The "Filters" dialog is displayed to specify filtering parameters for the alarm control object.

Alarms, Events and Trends

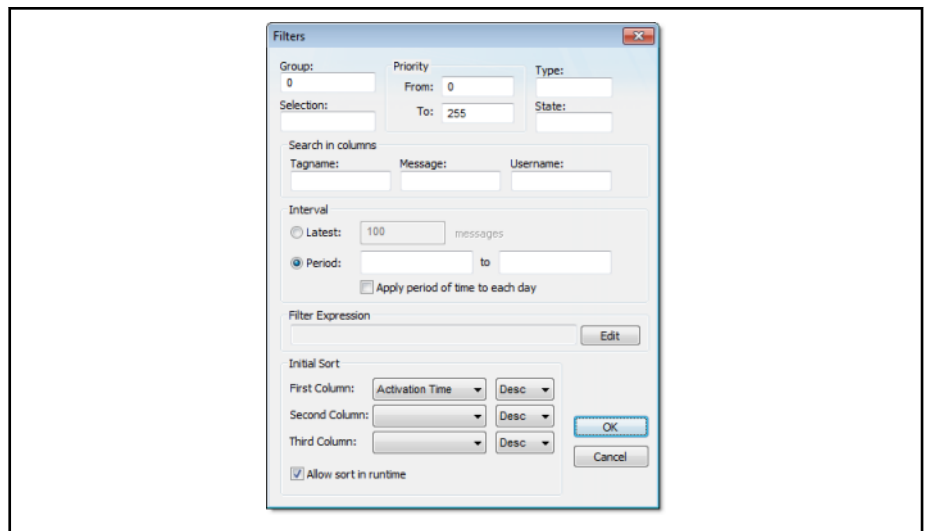


Fig. 9-13: Filters dialog

- Use the **Group** field to filter messages by the number of the [alarm worksheet on page 247](#). The worksheets are organized in the folder Alarms under the WinStudioTasks folder, starting with 1. If 0 is specified for the group, all worksheets are displayed. Commas or dashes can be used to specify an array of groups, for example, 1,3,5-6.
- Use the **Selection** field to filter messages by the selection text configured on the alarm worksheet.
- In the "Priority" pane, use the **From** and **To** fields to filter messages according to the priority configured on the alarm worksheet. Enter numerical values into these fields to limit the priority range.
- Use the **Type** field to filter messages by the alarm type (e.g., HiHi, Hi, Lo, LoLo, Rate, Dev+, Dev-). Commas can be used to specify more than one type; for example, "HiHi,LoLo".
- Use the **State** field to filter messages by the alarm status:

Value	Description
0	All alarms (default)
1	All active and unacknowledged alarms
2	All active and acknowledged alarms
3	All inactive and acknowledged alarms
4	All inactive and unacknowledged alarms

Leaving this field blank is effectively the same as entering a value of 0.

- In the "Search in columns" pane, use the **Tagname**, **Message**, and/or **Username** text fields to specify criteria for filtering messages. Enter a tag name, message, and/or user name into the text field to search for in "WinStudio".
- Use the parameters in the "Interval" pane to filter messages by the last x messages (**Latest**) or based on a period of time (**Period**). If no interval is specified, only the alarms for the current day are displayed.
- Use the "Filter Expression" pane to configure an expression to filter unwanted messages out of the display. Only messages that fulfill the expression are shown. To enter an expression, click on the **Edit** button. The dialog "Alarm Filter Expression" is displayed.

Alarms, Events and Trends



- String tags can be specified in curly brackets (e.g. {tag-name}) in the **Group**, **Selection**, **Tagname**, **Message**, and **Username** fields to change these values at runtime.
- String tags have to be specified without curly brackets in the **Type** field and the **Period** fields of the Interval pane. In these fields, values cannot be entered directly.
- Integer tags can be specified in the **From** and **To** fields in the "Priority" pane and the **State** field, and the **Latest** field in the "Interval" pane.
- Placeholder characters (* and ?) can be used when specifying values for the **Selection**, **Tagname**, **Message**, and **Username** fields.

- Use the Filter Expression pane to configure an expression to filter unwanted messages out of the display. Only messages that fulfill the expression are shown.

To enter an expression, click on the **Edit** button. The dialog "Alarm Filter Expression" is displayed.

The filter expression has to comply with the following basic syntax:

[Column Name]ComparisonOperator'Value'

where the **column name** is the name of a column in the Alarm/Event Control object.

For example:

[Activation Time]>'08/17/2007 15:00'

This filter only displays alarm messages with activation times greater (later) than 15:00 on 08/17/2007.



- The maximum number of characters is 1024 for engineering mode and 2048 for runtime mode.
- The **Display Value** and **State** columns are not supported by the filter expression.



- Several conditions can be combined simultaneously by using the logic operators AND, OR and NOT.

For example: **[Type]='HiHi' OR [Type]='LoLo' AND [Activation Time]>'08/17/2007 15:00'**

- In the filter expression, placeholder characters (* and ?) can be used.
- It is not necessary to use square brackets if the **column name** consists of only one word (e.g., **Value**).
- The filter expression can be changed at runtime by specifying String tags in curly brackets.

For example: **[Value]='{AlarmFilterValue}'**

- To use more than 1024 characters in the filter expression at runtime, more than one tag is required between curly brackets using the **{TagName1} AND {TagName2}** syntax.

Alarms, Events and Trends

- Use the parameters in the "Initial Sort" pane to set the default sorting order. Select a sort type from the **Column** combo box and select **Asc** or **Desc** to sort in ascending or descending order. Up to three levels of sorting can be configured.



If all three levels are configured with sort types other than activation time, the project automatically sorts on a fourth level according to activation time in descending order.

The type of this fourth-level sort cannot be changed, however, its default order can be toggled - from descending to ascending - by manually editing the project file (**project_name.app**) by changing the following setting:

[Objects]

DescendingAlarmListTime=TRUE | FALSE

TRUE sorts in descending order, **FALSE** sorts in ascending order. Please note that this setting only works for projects created with or updated to "Rexroth WinStudio v7.2" or later.

Click on the **Allow sort in runtime** checkbox to enable changing the sort order at runtime.

Advanced settings

Click the **Advanced** button to open the "Advanced" dialog to set advanced properties for the alarm object.

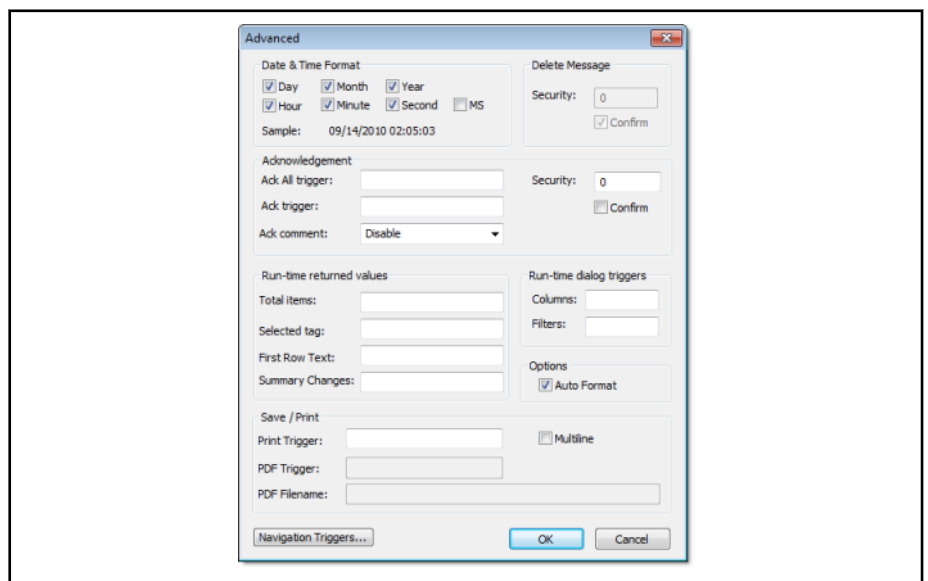


Fig. 9-14: Advanced dialog

- Use the parameters in the "Date & Time Format" pane to control which date and time information displays in the alarm message. Click on (enable) a checkbox to include that element in the display. Note: **MS** stands for milliseconds.



Please note the **Sample** text as preview how the information will look in the alarm message.

- Use the parameters in the "Acknowledgement" pane to control how alarms are acknowledged.

Alarms, Events and Trends

- **Security** field: Enter a numeric value to specify which security level can acknowledge an alarm message. Only users with the specified level can respond.
- **Ack All trigger** field: Enter a tag to receive a value. If the tag changes its value, it indicates that all messages in the alarm object have been acknowledged.
- **Ack trigger** field: Enter a tag to receive a value. If the tag changes its value, it indicates that the message at the top of the alarm object has been acknowledged.
- **Require confirmation** checkbox: Click on (enable) this box to display a confirmation dialog when the user tries to acknowledge a single alarm.
- **Enable comment (individual ack only)** checkbox: Click on (enable) this box to enable entering comments about the alarm, just after acknowledging it.
- Use the parameters in the "Runtime dialog triggers" pane to control the following:
 - **Columns** field: Enter a tag to receive a value. If the tag changes its value, a dialog is opened for customization of the columns that are visible in the object.
 - **Filters** field: Enter a tag to receive a value. If the tag changes its value, a dialog is opened to filter the columns that are visible in the object.
- Use the parameters in the "Delete Message" pane to control who can delete alarm messages from the alarm history:
 - **Security**: Use this field to specify which security level can delete alarm messages. Only users with the specified security level are allowed to delete alarm messages.
 - **Confirm**: Click on (enable) this box to require the user to confirm deleting a message before "WinStudio" actually deletes the selected alarm message.
- **Print Trigger**: If the tag configured in this field is toggled, the current state of the Alarm/Event Control object is sent to the default printer.
- **PDF Trigger** field: If the tag configured in this field is toggled, the current state of the Alarm/Event Control object is saved as a PDF file at the location specified by **PDF Filename**.
- **PDF Filename** field: Enter a complete file path and name where the PDF file is to be saved. Tag names using the **{tag}** syntax can also be entered.



PDF Trigger and **PDF Filename** are not supported in projects running on Windows Embedded or Thin Client.

- **Multiline** checkbox: If this option is enabled, the print output or PDF is formatted according to the available column space, and the text within each cell is wrapped so that it is completely displayed.
- **Total items** field: Enter an Integer tag to see how many alarms remain after "WinStudio" filters the alarm objects using parameters specified in the [Filters dialog on page 262](#).

Alarms, Events and Trends

- **Auto Format** checkbox: If enabled, decimal values in the Display Value, Previous and Value columns are formatted according to the virtual table created by the [SetDecimalPoints on page 604](#) function.
- **Selected tag** field: Enter a String tag to enable the end user to click on an alarm message to see the name of the tag associated with that alarm event.
- **First Row Text** field: Enter a String tag. This tag receives the text of all fields from the first row of the Alarm/Event Control. These fields are <tab>-delimited. Whenever the first row changes – either due to a new Alarm/Event or simply because the rows are reordered – the value of the configured tag is updated.
- **Total items** field: Enter an Integer tag. This tag receives the running count of the number of changes in the Alarm/Event Control. For example, if a new alarm occurs or an alarm is acknowledged, the value of the configured tag is increased. Reordering the rows is not counted as a change.
- Click on the **Navigation Triggers** button to open the following dialog:

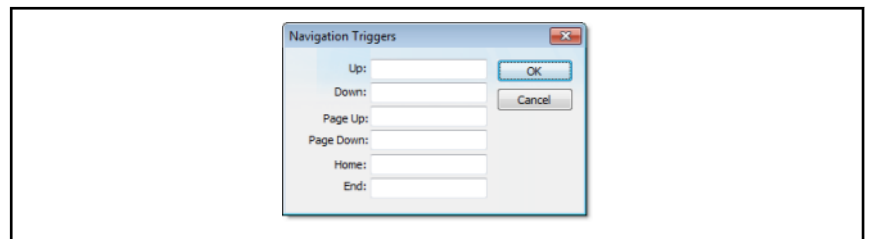


Fig.9-15: Navigation Triggers dialog

By configuration of tags in the respective fields, the functions scroll up, scroll down, page up, page down, home or end can be executed for displayed alarm objects. If the values of the configured tags change, the alarm control object navigates that way. This is useful for adding navigation control elements to a screen. For example, if the same tag is set to the **Up** field in this dialog as well as to a [pushbutton object on page 178](#), the alarm control object scrolls up in case the pushbutton object is pressed.

Finish the settings by clicking on **OK** to close the "Advanced" dialog.

9.5 Trend Worksheet

9.5.1 Trend Worksheet, General Information

The "Trend" folder enables configuring history groups that store trend curves. The trend worksheet can be used to specify tags with values that have to be stored on disk and to create history files for trend graphs. The project stores the samples in a binary history file (*.hst) and shows both history and online samples in a trend graph screen.

The trend worksheet is executed by the Background Task module. It manages saving of trend data in the history, however, does not display this data to the operator. To display trend data, the [trend object on page 281](#) in the "WinStudio Screen Tools" tool window has to be inserted into a WinStudio screen and configured.

To create a new trend worksheet, open the context menu of the "Tasks" folder and select "Trend" in the "Add" menu.

To edit an existing trend worksheet, double-click it in the Project Explorer.

Alarms, Events and Trends

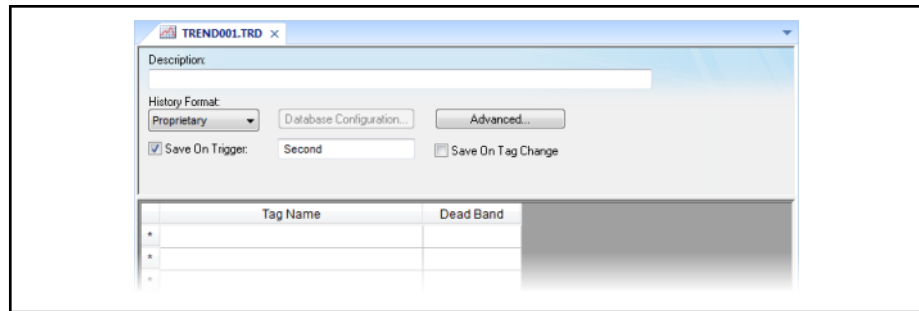


Fig. 9-16: Trend worksheet

The trend worksheet is divided into two areas:

- Header area (top section) containing information for the whole group
- Body area (bottom section) to define each tag in the group. This section contains several columns (only two are shown in the preceding figure).

Use the header parameters on this worksheet as follows:

- **Description** field: Enter a description of the worksheet for documentation purposes.
- **History Format**: Click to select a trend history format. The following options are available:
 - **Proprietary**
 - **File Format**: Binary
 - **Default path**: ...\\project_name\\Hst\\GGYYDDMM.HST, where:
 - **YY** = two last digits of the year
 - **MM** = month
 - **DD** = day



"WinStudio" provides the **HST2TXT.EXE** and **TXT2HST.EXE** programs to convert trend history files from binary (**.hst**) to text (**.txt**) and vice versa. For further information on these programs, refer to [Converting trend history files from binary to text on page 272](#) and [Converting trend history files from text to binary on page 273](#).

- **Database**
 - **Database type**: Selected by the user
 - **Default table name**: TRENDGGG (GGG = trend worksheet number; e.g., TREND001 for the trend worksheet 001)

For further information about the structure of the database table that "WinStudio" uses to save history files, refer to [Database interface on page 429](#).

NOTICE

String tags can be specified in many fields of the trend worksheet to change these values at runtime. However, this may influence how these values are stored in the trend history:

- If the history format is **Proprietary**, the value of the String tag is converted to a numerical value (if possible) and then saved to the history file. If numeric conversion is not possible, a value of 0 is saved.
 - If the history format is **Database**, the actual value of the String tag is saved in the database.
-
- **Database Configuration:** Opens the [Database Configuration on page 438](#) dialog, to enter the required settings to link the project to an external SQL relational database for the purpose of saving the trend history.
 - **Save On Trigger** checkbox and field: Click on (enable) the checkbox and type a tag name to save trend samples if someone changes the specified tag. (The tag change can be an event from the Scheduler.)
 - **Save On Tag Change** checkbox: Click (enable) to always save the trend sample in case of a value change in any of the tags from that group.
 - **Advanced:** Click to display the "Trend Advanced Settings" dialog. For information about completing the fields in this window, refer to [Batch history configuration on page 273](#).

Use the body parameters on this worksheet as follows:

- **Tag Name** field: Enter the variable name to be saved in the history file.
- **Dead Band** checkbox: Enter a value to filter acceptable changes when **Save On Tag Change** is used.

For example:

The dead band has a value of 5. If the tag value is set to 50 and changes to 52, the system does not register this change in the database as it is smaller than 5. If the change is at least 5, the new value is saved in the history file.

- **Field:** Name of the field in the database where the tag is stored. If this field is left blank, the name of the tag is used as the tag name. For array tags and classes the characters "[", "]" and "." are replaced by "_".

Examples:

Tag name	Default field
MyArray[1]	MyArray_1
MyClass.Member1	MyClass_Member1
MyClass[3].Member2	MyClass_3_Member2



The Trend task can accept only up to 240 tags in a single worksheet. If more than 240 tags are manually configured in the same worksheet, the Trend task generates an error on running the finished project.

Also refer to:

- [Converting trend history files from binary to text on page 272](#)

Alarms, Events and Trends

- [Converting trend history files from text to binary on page 273](#)
- [Creating batch history on page 273](#)
- [Configuring a default database for all task histories on page 442.](#)

9.5.2 Converting Trend History Files from Binary to Text

By default, "WinStudio" saves trend history files in a binary format (**.hst**). If these files are required in **.txt** format, "WinStudio" provides the **HST2TXT.EXE** program to convert trend history files from binary into text format.

To convert a file, proceed as follows:

1. From a DOS window, change directory (**cd**) to the "WinStudio" **\Bin** directory:

cd Bin

2. At the command prompt, copy the **Hst2txt.exe** into the same directory where the **.hst** file is located.
3. Enter **Hst2txt.exe** into the command prompt and specify the following parameters:

- **filename:** Name of the trend history file to convert
- **[separator]:** Data separator character (default is <TAB>)
- **[/e]:** Extended functionality (convert data with more than 10 characters)
- **[/f:HH:MM:SS]:** Start time in hours (**HH**), minutes (**MM**) and seconds (**SS**)
- **[/f:HH:MM:SS]:** End time in hours (**HH**), minutes (**MM**) and seconds (**SS**)
- **[/m]:** Include milliseconds in the **Time** column (Type **1** to print the milliseconds value in the text file created from the **.hst** file)

For example:

Hst2txt.exe 01952010.hst

The program creates a **.hdr** file (header area) and the **.txt** file. Both are plain text files that can be displayed in any text editor (e.g. Notepad).

- The **.hdr** file contains the name of the tags configured in the trend worksheet.
 - The **.txt** file contains the tag values saved in the history file.
4. After the program has converted the file, enter "Exit" to close the DOS window.



Alternatively, the [HST2TXT on page 590](#) calculation script can be used in a [math worksheet on page 324](#) to automatically convert binary files into text format without having to use a DOS window.

Also refer to:

- [Converting trend history files from text to binary on page 273](#)
- [Creating batch history on page 273](#)
- [Configuring a default database for all task histories on page 442.](#)

9.5.3 Converting Trend History Files from Text to Binary

"WinStudio" provides the **TXT2HST.EXE** program to convert text files back into binary format.

To convert a file, proceed as follows:

1. From a DOS window, change directory (**cd**) to the "WinStudio" \Bin directory:

cd Bin

2. In the command prompt, copy the **Txt2hst.exe** into the same directory where the **.txt** file is located.
3. Enter **Txt2hst.exe** into the command prompt and specify the following parameters:

- **filename:** Name of the ASCII file with history data to convert
- **[separator]:** Data separator character (default is <TAB>)
- **[/e]:** Extended functionality (data value with more than 10 characters)
- **[/f:HH:MM:SS]:** Start time of data value in hours (**HH**), minutes (**MM**) and seconds (**SS**)
- **[/f:HH:MM:SS]:** End time of data value in hours (**HH**), minutes (**MM**) and seconds (**SS**)

For example:

Txtt2hst.exe 02950201.txt

The program creates a **.hdr** ("header") file and converts the **.txt** file into a binary **.hst** file.

4. After the program has converted the file, enter "Exit" to close the DOS window.



For the **TXT2HST.EXE** program, no calculation scripts can be created for use in a [math worksheet on page 324](#) to convert text files into binary format as for **HST2TXT.EXE**. The math script shortcut is available for binary files only.

Also refer to:

- [Converting trend history files from binary to text on page 272](#)
- [Creating batch history on page 273](#)
- [Configuring a default database for all task histories on page 442](#).

9.5.4 Creating Batch History

"WinStudio" provides powerful tools to create and manage historical batch information. Batches can be created by using the following formats:

- **Proprietary:** When using the proprietary format, each batch will be stored on a different historical file. Historical data can be saved in both the normal historical file and batch files at the same time (refer to [Trend Folder on page 269](#) for further information about these files).
- **Database:** The historical data used for the batch is saved in the same table as the normal historical data. An additional table called Batch History contains tabs with the information about the batches. The following list describes the fields on the batch history table:

Alarms, Events and Trends

Field Name	Data type	Description
Group_Number	Integer	Trend group number. This is the number of the worksheet created to specify the tags that are to be stored in the batch history.
Batch_Name	String	Name of the batch
Start_Time	Time stamp	Date and time the batch was started.
End_Time	Time stamp	Date and time the batch was ended.
Pri_Table	String	Reserved
Sec_Table	String	Reserved
Description	String	Batch description
Deleted	Boolean	0: Batch has not been deleted 1: Batch has been deleted



The name of the table and the name of the columns created in the database can be customized by editing the **project_name.APP** file as follows:

[Trend]

DefaultName=NewName

[TrendGroupPRI|SEC]

BatchHistory=Tablename

For example:

[TREND001PRI]

BatchHistory=MyTableForPrimaryDB

[TREND001SEC]

BatchHistory=MyTableForSecondaryDB

[Trend]

Group_Number=Trend_Worksheet

Batch_Name=Load_Number

Batch history configuration

Add a trend worksheet (refer to [Trend folder on page 269](#)) and click on the Advanced button to display the following window:

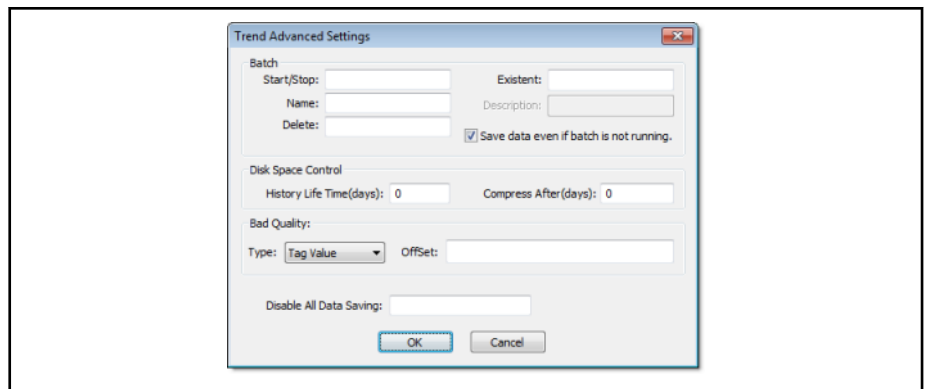


Fig. 9-17: Trend Advanced Settings

In the **Batch** pane, the saving process of the batch history can be configured:

- **Start/Stop (input):** Enter the tag that is to start/stop the batches. If the tag in this field is set to TRUE (different from 0), "WinStudio" will either start saving data to the batch file (if the Proprietary format is used) or add a new tab to the BatchHistory table on the database, indicating that a new batch has been started. Note that historical data is saved according to the configuration of the **Save On Trigger** and **Save On Tag Change** fields on the trend worksheet.
- **Name (input):** This field represents the batch name. The meaning of this field depends on the format selected on the trend worksheet:
 - If **Proprietary** is selected in the **Type** field, the **Name** is to comply with the format **[Path]<FileName>**, where:
 - **Path:** An optional field. If the path is not specified, the batch history file is stored under the same path as the **project_name.app** file.
 - **FileName:** Name of the batch history file.
 - If **Database** is selected in the **Type** field, the value in this field is stored in the **Batch_Name** field of the "BatchHistory" table.



In this field, tag names can be entered in curly brackets.

(e.g., C:\MyBatches\{MyTagWithName}\{MyTagWithNumber}.hst).

- **Delete (input):** If the tag specified in this field changes its value, the batch will be deleted. In **Proprietary** format, the batch history file will be removed. In **Database** format, the **Delete** field in the "BatchHistory" table will be set to true, however, the saved historical data remains. The Trend object only detects batches if the delete field is set to "0" (zero).
- **Existent (output):** The tag entered in this field will receive the value "1" if the batch specified in the **Name** field already exists. Otherwise, the tag will receive the value "0".
- **Description (output):** This field is only available for the **Database** format. If the tag in the **Start/Stop** field changes to TRUE, the tab that is added to the "BatchHistory" table displays the string in this field.



In this field, tag names can be entered between curly brackets (e.g., {MyTag}).

Alarms, Events and Trends

- **Save data even if batch is not running:** If this checkbox is disabled, historical data will be only saved if the tag in the **Start/Stop** field is TRUE.



Historical batch data can be displayed to the user either in graphical or table format.

Refer to [Trend folder on page 269](#) or [Grid object on page 305](#) to display information in these formats.

In the **Disk Space Control** pane, the use of the disk space can be controlled:

- **History Life Time (days)** field: Specify how many days the history file is kept on the disk. After the specified period, "WinStudio" automatically deletes the file. Use this option only for files based on a date.
- **Compress After (days)** field: Specify how many days the trend history file (*.hst) is kept on the disk before compressing the file. After the specified period, "WinStudio" automatically compresses the file. Use this option only for files based on a date. This option is not available for Windows Embedded target systems.

In the **Bad Quality** pane, the value to be actually saved in the batch history in case the tag quality is BAD can be specified.

Type	Description
Tag value	The actual value of the project tag if the tag quality was BAD, plus the specified Offset (if any).
Min value	The minimum historical value of the project tag, minus the specified Offset (if any).
Max value	The maximum historical value of the project tag, plus the specified Offset (if any).
Value	The specified Value only.
NaN	Not a number. Please note: If the history format is set to Database and Bad quality is set to NaN , all database fields are saved as Float type. Additionally, if a trend object is configured to use the history created by this trend worksheet, NaN entries are counted as 0 for the calculation of statistic average values and trend deviations.



The **Bad Quality** option cannot be used in projects running on Windows Embedded target systems.

Finally, enter the name of a tag into the **Disable All Data Saving** field. When the value of the tag is TRUE (non-zero) during runtime, data saving is disabled for this worksheet. Other trend worksheets are not affected.

9.5.5 Setting the Trend Database

1. Open a trend worksheet by double-clicking on an element in the "Trends" folder.

Alarms, Events and Trends

	Tag Name	Dead Band
1	TagSin	
2	TagCos	
3	TagRamnLn	

Fig.9-18: Trend worksheet example

- Click on the arrow to the right of the **History Format** field. Select **Database**.
- Click the **Database Configuration** button. The [Database Configuration dialog on page 438](#) is displayed. Enter the applicable data in this window. To finish the process, click on **OK**.
- Click on the **File** selection menu and on **Save** to save the changes to the "Trend".

Trend table on a relational database

The fields saved in the trend history are described in the following table:

Field name	Data type	Information
Time_Stamp	Time stamp	Time stamp (date and time) when the data was saved.
<TagName>	Integer or Real (depending on the tag type)	"WinStudio" will create one field (column) in the database for each tag configured in the trend worksheet.

9.6 Trend Display

9.6.1 Trend Display, General Information

The main functions of the trend control object are:

- Display of multiple trend curves simultaneously
- Support of different data sources, such as tag, batch, database and text file
- Capability to generate X/Y graphs from the configured data sources (please refer to [Appendix A on page 298](#) for an example of an X/Y chart).
- Simultaneous display of an unlimited number of data points. This function might be limited by the hardware used since the available memory and performance may vary.
- Built-in toolbar, which provides interfaces to interact with the trend control object at runtime
- Built-in legend, which displays the main information associated with each pen linked to the object
- Zooming and auto-scaling tools
- Horizontal and vertical orientation

Alarms, Events and Trends

9.6.2 Trend Display Operating Elements

At runtime, trend displays have built-in operating elements that can be used to change how trends are displayed. In the following, the main operating elements and their function are described.

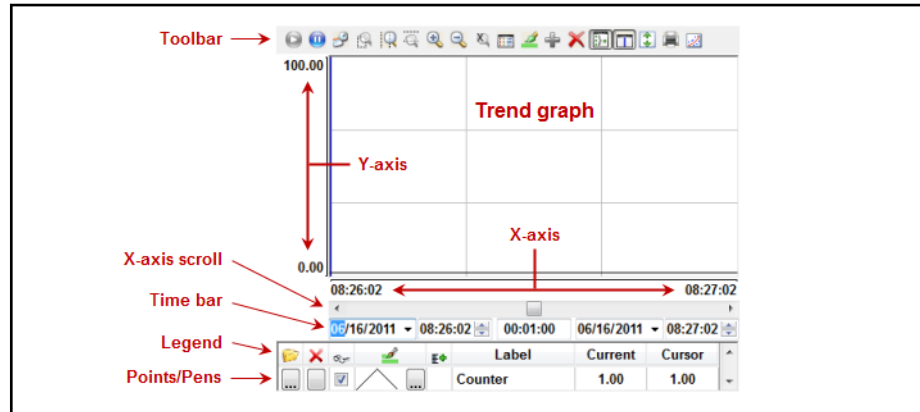


Fig.9-19: Trend display operating elements

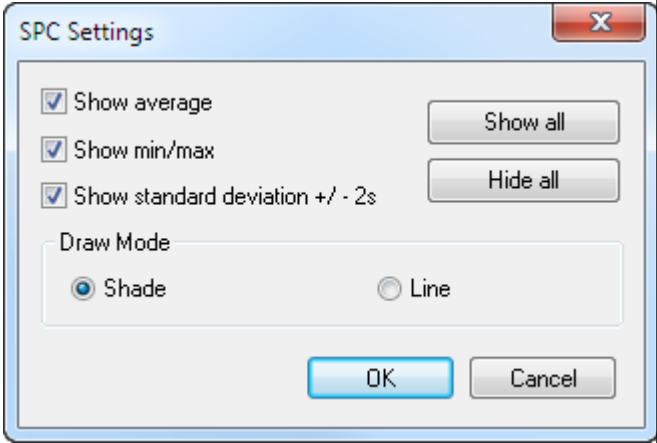
Toolbar

Command	Symbol	Description	Activation tag
Run		Sets the trend object to Run mode (a.k.a. Online mode). In this mode, the X-axis continues to scroll with the passage of time and the trends are updated with current tag values.	0 = Run mode on 1 = Run mode off
Stop		Sets the trend display to Stop mode (a.k.a. Historical mode). In this mode, the X-axis is stopped and the trends display only historical data. If decimation is enabled for one or more trends, calculation and redrawing only takes place in this mode.	0 = Stop mode on 1 = Stop mode off
Period		Opens a dialog to modify the main settings for the X-axis scale.	If the activation tag changes its value (e.g., toggles), this command is executed.
Win-dow zoom		Allows the user to click on the trend graph and drag the cursor to select the area that has to be visible when the cursor is released. This option is disabled if the Multiple Section option (for the Y scale) is active.	
Horizontal zoom		Allows the user to click on two points in the trend graph, defining the horizontal scale that has to be available.	
Vertical zoom		Allows the user to click on two points in the trend graph, defining the vertical scale that has to be available. This option is disabled if the Multiple Section option (for the Y scale) is active.	0 = Zoom in on 1 = Zoom in off
Zoom in		Enables zooming in when clicking on the trend graph.	
Zoom out		Enables zooming out when clicking on the trend graph.	0 = Zoom out on 1 = Zoom out off

Alarms, Events and Trends

Command	Symbol	Description	Activation tag
Cancel zoom		Cancels the current Window , Horizontal or Vertical zoom and returns the trend graph to its original scale.	If the activation tag changes its value (e.g., toggles), this command is executed.
Legend properties		Opens a dialog to modify the legend main settings.	
Pen style		Opens a dialog to modify the pen style of the selected trend.	
Add pen		Opens a dialog to add a new trends to the trend display.	
Remove pen		Removes the selected trend from the trend display	
Multiple sections		Switches the Y scale between multiple sections (a section for each trend) and single section (all trends share the same Y scale section).	0 = Multiple sections on 1 = Multiple sections off
Cursor		Switches the cursor (ruler) between visible and hidden.	0 = Cursor on 1 = Cursor off
Auto scale		Changes the Y axis scale to fit all values from the trends that are currently being monitored.	If the activation tag changes its value (e.g., toggles), this command is executed.

Alarms, Events and Trends

Command	Symbol	Description	Activation tag
Print		Prints the current state of the trend control. (Historical data is not printed.)	If the activation tag changes its value (e.g., toggles), this command is executed.
SPC		Opens a dialog to display the statistical process control (SPC) information for the selected trend:  <i>Draw mode...</i> Shade: Draws the average value as a dashed line and the min/max values and standard deviation as shaded areas. Line: Draws the average value and standard deviation as dashed lines and the min/max values as solid lines. Show average: Show the calculated average of all historical trend values. Note: If a value is not a number (NaN), for example, if a tag is marked as BAD quality, it is counted as 0 for calculating the average. Show standard deviation: Shows the standard deviation of the trend. A low standard deviation indicates that the actual value tends to stay close to the average. A high standard deviation indicates that the actual value tends to vary greatly from the average. Show min/max: Shows the minimum and maximum historical values of the trend.	If the activation tag changes its value (e.g., toggles), this command is executed. Additionally, the bit properties of the tag (B0–B4) can be used to pre-select options in the dialog: tagname->B0 0 = Close dialog 1 = Open dialog tagname->B1 0 = Draw mode: Line selected 1 = Draw mode: Shade selected tagname->B2 0 = Show average cleared 1 = Show average selected tagname->B3 0 = Show standard deviation cleared 1 = Show standard deviation selected tagname->B4 0 = Show min/max cleared 1 = Show min/max selected



Activation tags are configured in the Object Properties of the trend object. For further information, refer to [Toolbar on page 291](#).

Legend

Command	Symbol	Description
Selection		Opens a dialog to replace the data point associated with the selected trend on the legend.
Remove		Removes the selected trend from the trend display
Hide		If this option is enabled, the selected trend is invisible. Otherwise, it is displayed.

Command	Symbol	Description
Pen style		Opens an embedded dialog to modify the pen style of the selected trend.
Scale		If this option is enabled, the Y axis scale is visible. Otherwise, it is hidden. The scale can only be hidden if the Multiple Sections option is disabled.

9.6.3 Trend Object Properties

Trend Object Properties, General Information

Accessing the dialog

The "Object Properties: Trend object" dialog is used to configure the basic properties of trend objects.

To access the "Object Properties" dialog for a specific screen object, proceed as follows:

- Open the WinStudio screen with the trend object to be configured in the editor
- Right-click on the screen object and select **Properties** in the context menu
- Double-click on the screen object.

The dialog in detail

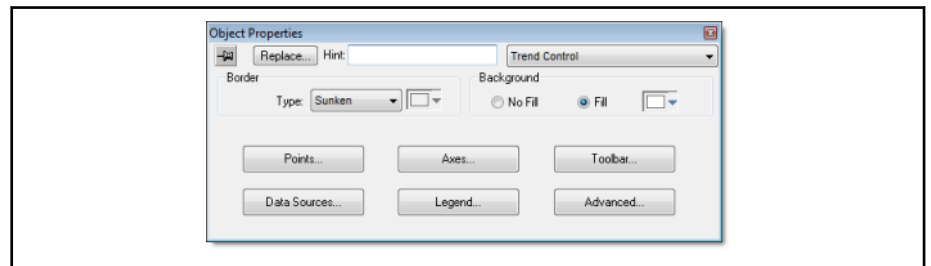


Fig.9-20: Object Properties dialog: Trend object

In addition to the elements that are similar in all Object Properties dialogs, the "Object Properties: Trend object" dialog provides the following elements:

Pane / Element name		Description
Border	Type	Sets the type of border around the graph area of the trend display. (There are no borders around the legend or toolbar of the trend display)
	Color	Sets the color of the border, if the border type is set to Solid . For further information, refer to Selecting colors and fill effects.
Background	NoFill/Fill	Enables the background fill for the graph area of the trend display. (There are no backgrounds for the legend or toolbar of the trend display.) If the fill is disabled, the graph is transparent to for any other screen object behind the trend display.
	Color	Sets the color and fill effect of the background fill if enabled. For further information, refer to Selecting colors and fill effects.
Points		Opens the Points dialog to configure the data points (or pens) of the trend display. For further information, refer to Trend object: Points dialog on page 282
Axes		Enables configuration of the X and Y axes of the trend display as well as the horizontal or vertical position of the axes. For further information, refer to "Trend display: Axes"
Toolbar		Enables configuration of the user toolbar that is displayed above the trend display. For further information, refer to "Trend display: Toolbar"
Data sources		Enables configuration of multiple data sources for the trend display. For further information, refer to "Trend display: Data Sources"

Alarms, Events and Trends

Pane / Element name	Description
Legend	Enables configuration of the legend that is displayed below the trend display. For further information, refer to "Trend display: Legend"
Advanced settings	Enables configuration of the advanced properties of the trend display, e.g. runtime options and tag triggers. For further information, refer to "Trend display: Advanced"

Although trend objects support flexible configurations to fulfill specific project requirements, most of the settings are set by defaults based on the most common functions. Therefore, in many cases, only data points are configured (displayed at runtime) by simply clicking on the **Points** button in the "Object Properties" window.

Points

Points Dialog, General Information

The "Points" dialog is used to configure the data points for a trend display screen object. The value of each data point is represented as a pen in the trend display. It can be changed dynamically whether a data point is visible at runtime, regardless of how many data points are associated with the screen object.

Accessing the dialog

To access the "Points" dialog for a specific trend display screen object, first select the [Object Properties dialog for the screen object on page 281](#) and click on **Points**.

The dialog in detail

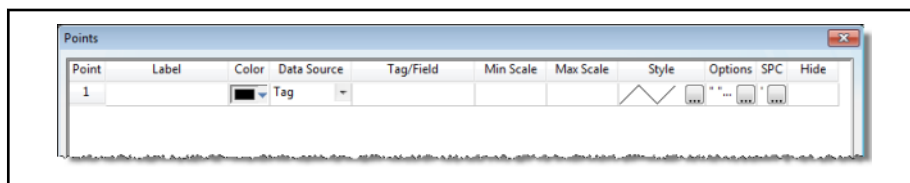


Fig.9-21: Object Properties dialog: Trend display

The following table summarizes the properties of each data point:

Column name	Description
Point	A unique ID number for the point, which is assigned automatically when the point is created in this interface.
Label	The label associated with the point can be displayed on the legend at runtime, providing a short reference to for each point.
Color	The color of the pen used to draw the values of the point on the trend display.
Data source	The data source for this point. Tag is available by default. All other sources have to be configured in the "Data Sources" dialog.
Tag/Field	The meaning of this parameter depends on the Data Source type associated with the data point. • Tag: Enter the name of the tag with values to display. If the tag is configured in the trend task, the history data is automatically retrieved; otherwise, only the online values are displayed. • Batch: Enter the name of the tag with values to be retrieved from the Batch History file generated by the trend task and displayed on the object. • Database: Enter the name of the field (column) in the SQL relational database that holds the data point values. • Text File: Enter the number of the column that holds the point values. The number 0 refers to the first column, 1 refers to the second column, etc.

Alarms, Events and Trends

Column name	Description
Min scale/Max scale	The scale of the Y-axis for this point. This overrides the default scale that is set in the Axes dialog on page 287 . Note: The Min scale and Max scale properties can contain numeric Real values with up to six decimal places. If higher precision is required, the Min scale and Max scale properties have to be configured with Real tags and the values have to be stored in those tags.
Style	The Line and marker style for this point. Click on the  button to open the Pen Style dialog on page 283 .
Options	Additional options for this point. Click on the  button to open the Options dialog on page 285 .
SPC	Calculated statistics to be used in statistical process control (SPC). Click on the  button to open the SPC dialog on page 286 .
Hide	Tag trigger: If the value is TRUE, the data point is hidden in the trend display.

Pen Style

In this dialog, the style of the pen used to draw the data point values on the object at runtime can be configured.

Accessing the dialog

To access the "Pen Style" dialog for a specific data point, open the "Points" dialog and click on the **Style** column for the data point.

It can also be opened at runtime to customize these settings on-the-fly.

The dialog in detail

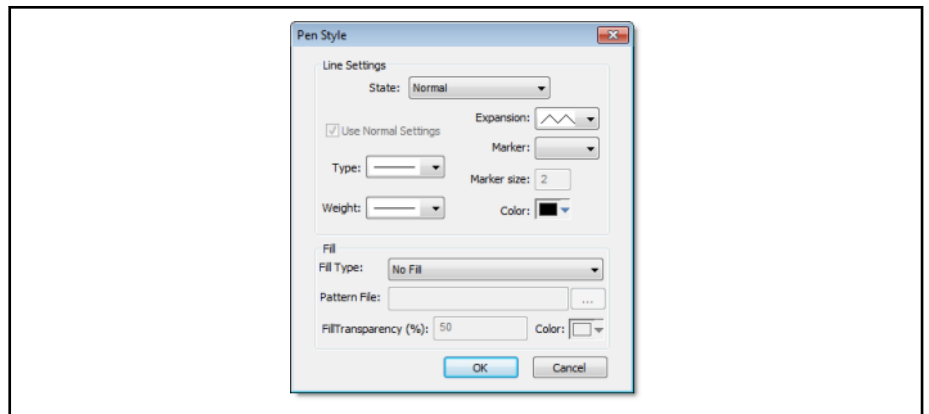


Fig. 9-22: Pen Style dialog

The "Pen Style" dialog includes the following elements:

Alarms, Events and Trends

Elements in the "Pen Style" dialog

Pane / Element name		Description
Line Settings	State	With the Options dialog a Hi Limit and a Lo Limit for each data point can be defined. The Pen Style dialog enables configuring different settings for the pen (e.g., color), both when its values are within the limits (Normal state) and not within the limits (Out of Limits state).
	Use Normal Settings	Available only for the Out of Limits state. If this option is enabled, the pen will always be displayed with the settings for the Normal state, even if the data point values are not within the limits configured for it.
	Type	The type of line (e.g., solid, dashed, dotted) that connects the data points.
	Weight	The weight of the line that connects the data points.
	Expansion	The algorithm used to connect the points, as follows:  : Consecutive points are directly connected to each other by an analog line. This option is suitable for numerical values.  : Consecutive points are connected only through horizontal or vertical steps (depending on the orientation of the trend display). This option is suitable for Boolean values.
	Marker:	The shape used to mark each data point. If no shape is selected, only the connecting line between points is displayed.
	Marker size	The size of the data point marker.
Fill	Color	The color of the trend line and data point markers.
	Fill type	The type of fill between the trend line and the number line.
	Pattern file	The graphics file used to fill the trend area. Available only if the Fill Type is set to Custom pattern . Click the browse button to open a Windows file browser and select the desired graphics file. The file should be located in the project folder. The following example shows trends with custom fill patterns.
	Color	The color used to fill the trend area. Available only if the Fill Type is set to Solid color .
	Fill Transparency (%)	The transparency level of the fill. (If the fill is transparent, other trends behind it can be seen through it, making the entire graph easier to read.) Available for both Custom pattern and Solid color .



When viewing the project on either a Windows Embedded device or a Thin Client (any OS), the Pen Style dialog available during runtime only enables changing the pen color.

Alarms, Events and Trends

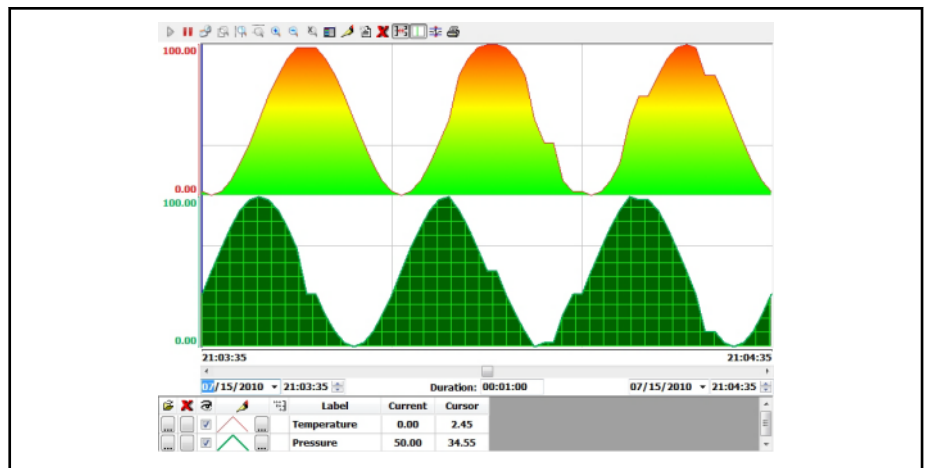


Fig.9-23: Trends with custom fill patterns

Options

Accessing the dialog

To access the "Pen Style" dialog for a specific data point, first access the "Points" dialog and then click the **Options** column for that data point.

The dialog in detail

Fig.9-24: Options dialog

The "Options" dialog includes the following elements:

Element name	Description
Description	This text can be displayed in the legend, providing a short description about the data point at runtime. When tags are used the default description is the one configured for the tag.
Eng. units	This text can be displayed in the legend, providing the engineering unit (i.e., the unit of measurement) associated with the data point at runtime. If tags are used the default units are the ones configured for the tag.
Low limit:	If the data point value is below this limit its pen can be displayed with a different style (e.g., color) at runtime. For further information, refer to Pen Style dialog on page 282 . If tags are used the default Low Limit is the Low Alarm value configured for the tag.
High limit	If the data point value is above this limit its pen can be displayed with a different style (e.g., color) at runtime. For further information, refer to Pen Style dialog on page 282 . If tags are used the default High Limit is the High Alarm value configured for the tag.
Hide scale	In this field, a tag can be configured to control the visibility of the scale (Y axis) associated with this pen at runtime by changing the value of this tag (0=Show; 1=Hide).

Alarms, Events and Trends

Element name	Description
Break interval	Maximum interval between two consecutive points. If the time between two consecutive samples is higher than this number (in seconds) the trend control assumes that there was no data collection in this period and does not draw a line linking both samples. If the X-axis is configured as numeric the value on this field represents a numeric scalar value. If the X-axis is configured as date/time the value for this field is given in seconds. This field has some special values: • -1: Do not connect the points. • -2: Connect only points in ascending order.
X-axis off set:	Offset for this data point from the X-axis scale configured for the object. This option is useful to display data from two or more data points using a different X scale (period of time/value) for each one to compare them. This enables comparing them. If the X-axis is configured as numeric the value on this field represents a numeric scalar value. If the X-axis is configured as date/time the value for this field is given in seconds.
Cursor value	Enter the name of a project tag. At runtime, the trend object updates the value of this tag with the value of the intersection between the data point pen and the vertical cursor (if any).
Y-axis log base	Enter a tag name or numerical value. If the value is 1 the Y-axis of the trend is displayed on a logarithmic (e.g., 1, 10, 100, 1000) rather than linear scale.
Annotation ID	N/A
Style modifier	N/A
Draw mode	Enter a tag name or numerical value. If the value is 1 the historical data for this trend is decimated before the trend is drawn. This means, the X-axis of the trend is divided into a number of intervals (determined by Max points in the Advanced on page 295 settings) and all of the data points within each interval are averaged together to be drawn as a single point. This is similar to the Decimation feature in the Advanced settings, except that the decimation is done only for this trend rather than for all trends in the trend display.

SPC

In the "SPC" dialog, project tags can be specified that will receive certain statistical values that are calculated from the entire data point history. These statistics are used in statistical process control (SPC), which is a method for monitoring processes and ensuring that they operate efficiently.

Accessing the dialog

To access the "SPC" dialog for a specific data point, first open the [Points dialog for the trend control screen object on page 282](#) and click on the **SPC** column for that data point.

The dialog in detail

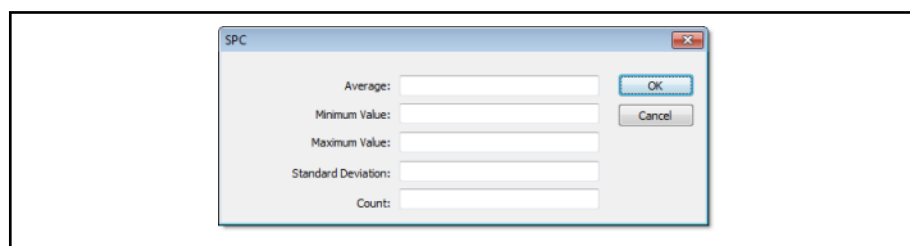


Fig. 9-25: Dialog: Trend Object properties

The "SPC" dialog includes the following elements:

Alarms, Events and Trends

Element name	Description
Average	Project tag (Real type) that receives the calculated average of all of the historical values of the data point. If a value is not a number (NaN) – for example, if a tag is marked as BAD quality – it is counted as 0 for the purpose of calculating the average.
Minimum value	Project tag (Real type) that receives the minimum historical value of the data point.
Maximum value	Project tag (Real type) that receives the maximum historical value of the data point.
Standard deviation	Project tag (Real type) that receives the standard deviation of the data point. A low standard deviation indicates that the value of the data point tends to stay close to the average. A high standard deviation indicates that the value tends to vary greatly from the average.
Count	Project tag (Integer or Real type) that receives the total number of historical values or samples for the data point. The count increases as the project runs and the historical database grows.

Axes

Accessing the dialog

To access the "Axes" dialog for a specific trend object, first select the [Object Properties dialog for the screen object on page 281](#) and click on **Axes**.

The dialog in detail

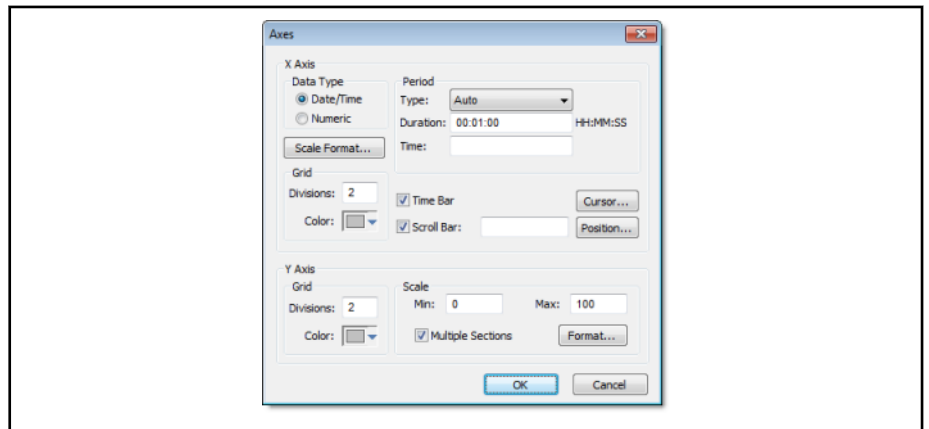


Fig. 9-26: Axes dialog

The "Axes" dialog contains the following elements:

Alarms, Events and Trends

Pane / Element name			Description
X Axis (1)	Data Type	Date/Time	
		Numeric	
		Scale For- mat	
	Period (if Data type is set to Date/ Time)	Type	• Auto: If this option is selected the trend control object works with Start Date/Time when triggered to Pause mode and works with Time Before Now when triggered to Play mode. • Start Date/Time: If this option is enabled, the value of the tag configured in the Time field defines the starting date/time for the data displayed on the object. • Time Before Now: If this option is enabled, the value of the tag configured in the Time field defines the time before the current date/time, which is used as the starting date/time for the data displayed on the object.
		Duration	Defines the period of data displayed on the object. A String tag can be configured in this field to change the duration dynamically at runtime by changing the value of this tag. The format of the value supported by this property is HH:MM:SS. For example, "36:00:00" (thirty six hours).
		Time	This field is optional. The value of the tag configured in this field represents a period of time, rather than a specific date or time. The meaning of this value depends on the option set for under Type. If the Time field is left blank (or if the tag configured in this field has the value "0"), the object displays data up to the current date/time. • If Type is set to Start Date/Time, the value of the tag configured in this field has to comply with the date/time format. For example, "02/10/2005 18:30:00". • If Type is set to Time Before Now, the value of the tag configured in this field has to comply with one of the following formats: Time (String value) e.g. "48:00:00" (forty eight hours). Number of hours (Real value) e.g. "2.5" (two hours and thirty minutes). If the Time field is left blank (or if the tag configured in this field has the value 0), the object displays data up to the current date/time.
	Period (if Data Type is set to Nu- meric)	Min/Max	Minimum and maximum values displayed on the X-axis. The Min and Max properties can hold real numeric values up to six decimal places. If higher precision is required, the Min and Max properties have to be configured with Real tags and the values have to be stored in these tags.
		Eng. units	Engineering Unit (e.g., kg, psi) that is associated with the X-axis at runtime.

Alarms, Events and Trends

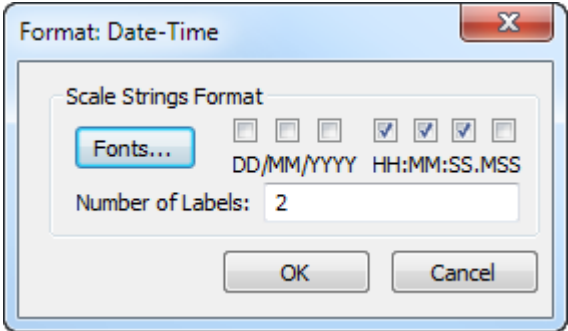
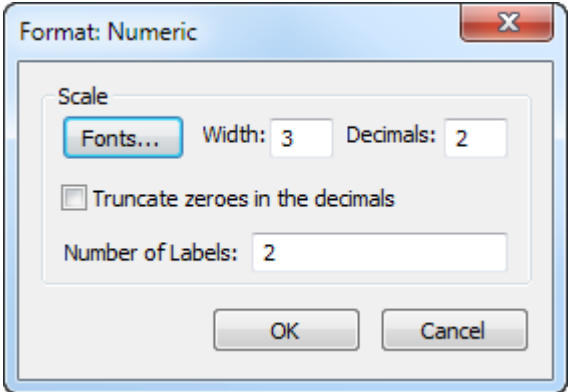
Pane / Element name			Description
X Axis (2)	Grid	Divisions	The number of divisions (vertical or horizontal lines) drawn on the object can be configured for the X- and/or Y-axis respectively as well as the color of these lines.
		Color	
		Time bar	If enabled, the time bar is displayed below the X-axis at runtime; otherwise, it is hidden. The time bar is a standard interface that can be used by the operator to change the X-axis scale at runtime.
		Scroll Bar	If checked, the scroll bar is displayed below the X-axis at runtime; otherwise, it is hidden. The scroll bar is a standard interface that can be used by the operator to navigate through the X-axis scale at runtime. Optionally, a tag can be configured in the Scroll Bar field to define the period for the scroll bar. If this field is left empty, the period is equal to the current value for duration of the X-axis.
		Cursor	The cursor is an optional ruler orthogonal to the X-axis that can be used at runtime to obtain the value of any pen at a specific point (intersection of the pen with the cursor). Press this button to display the Cursor dialog to configure the settings for the optional vertical cursor as follows:
		Position	Defines the position of the X-axis as well as its direction and orientation, as follows:
Y-axis	Grid	Divisions	The number of divisions (vertical or horizontal lines) drawn on the object can be configured for the X- and/or Y-axis respectively as well as the color of these lines.
		Color	
	Scale	Min/Max	Default minimum and maximum values displayed in the Y-axis. Used if more than one pen share the same scale (Multiple Sections disabled) and/or for points whose Min and Max fields are not configured (left blank).
		Multiple sections	If enabled, the Y-scale is automatically divided into one section for each pen; otherwise, all pens share the same Y-scale.
		Format	Opens a dialog for configuring the format of the labels displayed by the Y-axis.



The tags configured in the **Duration/Range** fields are automatically updated if the X-scale is changed dynamically at runtime, using the time bar embedded in the object.

- **Data type:** According to this setting, the X-axis can display either date/time values or numeric values.

Alarms, Events and Trends

Data type	Scale format
Date/Time	 The dialog box 'Format: Date-Time' contains a 'Scale Strings Format' section with a 'Fonts...' button and four checkboxes. Below these are the format strings 'DD/MM/YYYY' and 'HH:MM:SS.MSS'. The 'Number of Labels' is set to 2. 'OK' and 'Cancel' buttons are at the bottom.
Numeric	 The dialog box 'Format: Numeric' contains a 'Scale' section with a 'Fonts...' button, 'Width: 3', and 'Decimals: 2'. There is a checkbox for 'Truncate zeroes in the decimals' and the 'Number of Labels' is set to 2. 'OK' and 'Cancel' buttons are at the bottom.



The number of decimal points for the X-or Y-scale (decimals) can be configured with a tag. Therefore, this setting can be modified dynamically at runtime.

- Cursor:** The cursor is an optional ruler orthogonal to the X-axis that can be used at runtime to obtain the value of any pen at a specific point (intersection of the pen with the cursor). Press this button to display the Cursor dialog to configure the settings for the optional vertical cursor as follows:

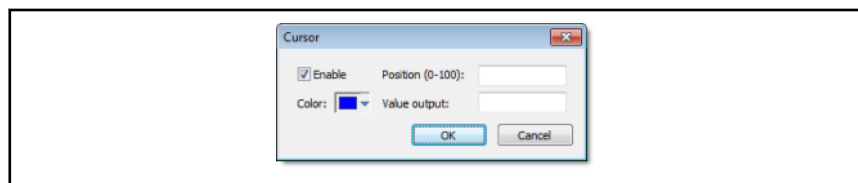


Fig.9-27: Cursor dialog

Property	Description
Enable	If enabled, the vertical cursor is visible at runtime.
Color	Color of the line drawn for the cursor.
Position (0-100)	In this field, a numeric tag can be configured, which is proportional to the position of the cursor on the X-axis, from 0 to 100%. If this value is changed the position of the cursor is automatically modified.
Value output	In this field, a String tag can be configured that returns the value of the X-axis in which the cursor is currently positioned.

Alarms, Events and Trends

- **Position:** Defines the position of the X-axis as well as its direction and orientation, as follows:

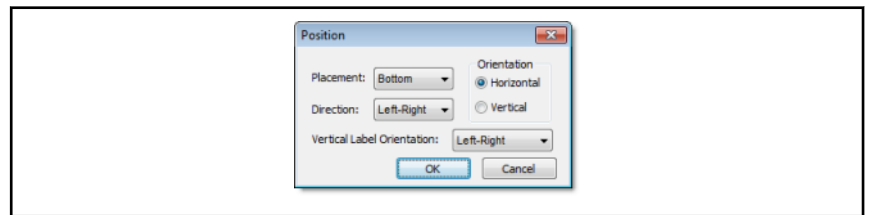


Fig.9-28: Position dialog

Property	Description
Placement	The side of the trend control on which the X-axis is placed.
Direction	Direction of the X-axis.
Orientation	Orientation of the X-axis.
Vertical orientation	The orientation of the text labels on the vertical axis, regardless of whether the vertical axis is X or Y.

Toolbar

Accessing the dialog

The "Toolbar" dialog is used to adjust the toolbar in a trend object.

To access the "Toolbar" dialog for a specific trend object, first select the [Object Properties dialog for the screen object on page 281](#) and click on **Toolbar**.

The dialog in detail

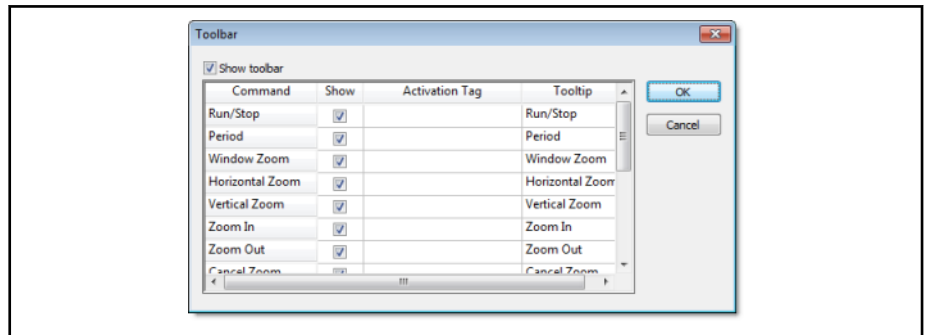


Fig.9-29: Toolbar dialog

The **Show toolbar** option controls whether the entire toolbar is shown at run-time. The toolbar can be hidden to save space or to prevent users from changing the trend display.

The commands/tools on the toolbar have the following properties:

Column name	Description
Command	The name of the command/tool. For further information about each tool, refer to
Show	The option to display the tool on the toolbar.
Activation tag	An optional tag trigger – if the value of the tag changes from FALSE (0) to TRUE (any non-zero value), the command is activated as if the operator clicked on the tool. This function can be used to script changes in the trend display at runtime.
Tooltip	The tooltip that is displayed when the mouse cursor is moved over the tool.

Alarms, Events and Trends

Data Source

Accessing the dialog

To access the "Data Sources" dialog for a specific trend object, first select the [Object Properties dialog for the screen object on page 281](#) and click on **Data Sources**.

The dialog in detail

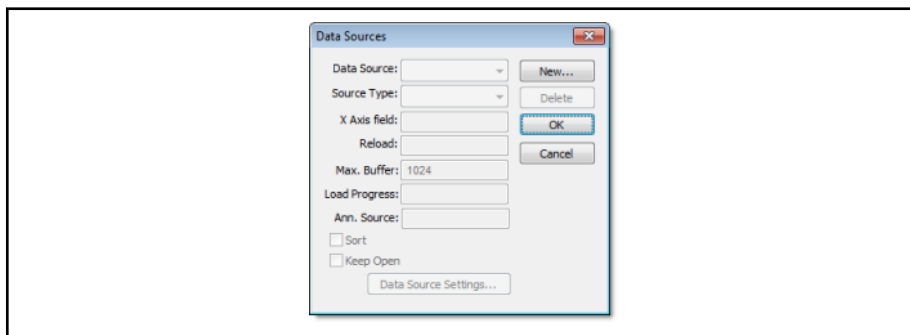


Fig.9-30: Object Properties dialog: Trend object

The **Data Source** defines the location of the values of the data points associated with it. Multiple points can share the same data source – it is not necessary to create a data source for each data point.

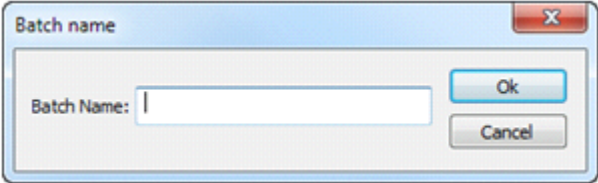
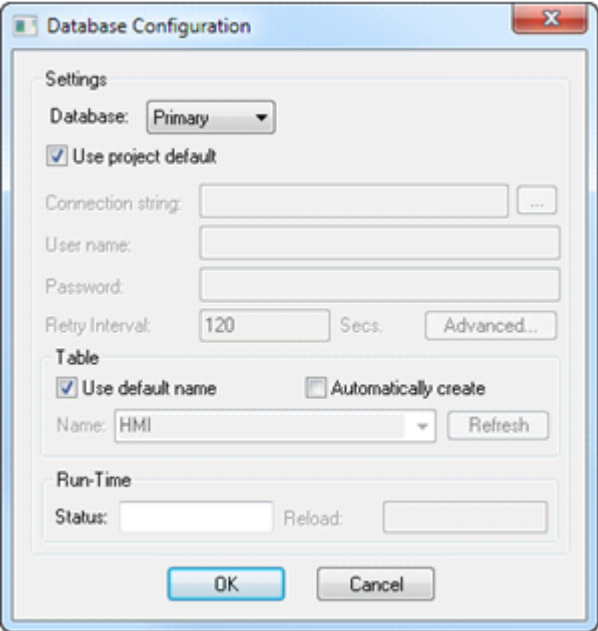
The data source tag is available by default to the trend object. Additional data sources can be added with the **New** button. The name entered is used as an alias to link the data points to this new data source.

The other fields in this dialog enable editing the data source settings:

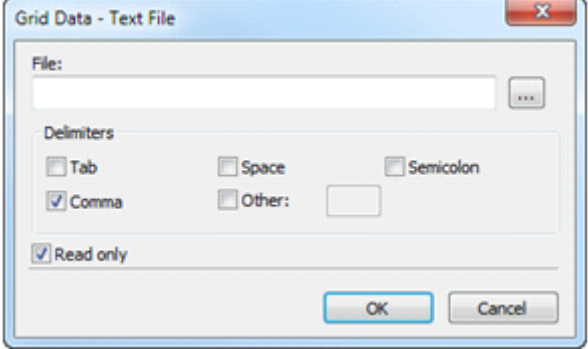
- **Source Type:** Select the source type of the location of the data point values.
- **X Axis field:** Enter the name of the field (column) of the data source that holds the X-axis data.
- **Max. Buffer:** The maximum amount of data (in bytes) that is held in the runtime memory.
- **Load Progress:** The tag in this field receives a real value (0-100) that represents the percentage of the data source load progress.
- **Sort:** This option is useful for plotting data from a text file. If this option is enabled (checked), the data is sorted and the Cursor column value is displayed until the **Max. Buffer** is full. If this function is disabled (unchecked) the data is not sorted and the Cursor column value is not displayed.
- **Keep Open:** This option keeps the data source open as long as the screen that contains the trend display is open. This increases the performance of the runtime project. However, keeping the data source open may cause other problems like database connection errors (if **Source Type** is set to **Database**) and file write conflicts (if **Source Type** is set to **Batch** or **Text File**). To close the data source after the data has been loaded, disable (uncheck) this option.
- **Data Source Settings:** Click to define the settings for the selected **Source Type**.

The following table summarizes the settings for each **Source Type**:

Alarms, Events and Trends

Data-source type	Description	X-axis field	Data source settings
Batch	Batch generated by the Trend task of "WinStudio"	Disabled. The X-axis data is automatically retrieved at the correct position of the proprietary batch file generated by "WinStudio".	 Enter the data point values in the Batch name field for their retrieval. In this field, tags can be configured in curly brackets to change this setting dynamically at runtime.
Database	SQL relational database	Field name that contains the X-axis data	 Configure the settings to link this data source to the SQL relational database that holds the data point values. For further information about this dialog interface, refer to Database configuration dialog window on page 438. Please refer to Appendix B on page 301 for an example of configuring databases.

Alarms, Events and Trends

Da-tasource type	Description	X-axis field	Data source settings
Text File	Text file (e.g., CSV file) with data point values separated by a specific delimiter	Number of the column that holds the X axis data. The number 0 refers to the first column, 1 refers to the second column, etc.	 Enter the name of the text file that holds the data points. The default path is the current project folder. In this field, a tag in curly brackets can be configured to dynamically change this setting at runtime. Additionally, one or more delimiters can be selected for the data saved in the text file. The value of each line is written in the text file between two delimiters. When using a comma as a delimiter, the grid object can read data from CSV files. Also custom delimiters can be selected by selecting the Other option. Please refer to Appendix A on page 298 for an example of configuring text files.

Legend

Accessing the dialog

To access the "Legend" dialog for a specific trend object, first select the [Object Properties dialog for the screen object on page 281](#) and click on **Legend**.

The dialog in detail

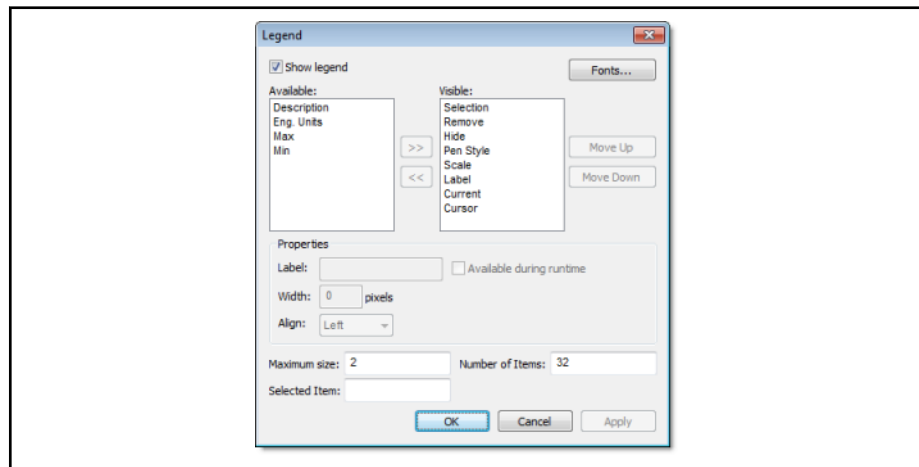


Fig.9-31: Legend dialog

The "Legend" dialog contains the following elements:

- **Show legend:** If enabled, the embedded legend is displayed at runtime. This interface provides useful information regarding the currently used pens.
- **Available/Visible:** The items in the **Visible** box are displayed in the legend at runtime. Items can be added to and removed from the **Visible** field using the » and « buttons respectively. Additionally, the **Move Up**

Alarms, Events and Trends

and **Move Down** buttons can be used to change the order in which the items are displayed in the legend at runtime. The following table lists the available legend items:

Element	Legend icon	Description
Eng. units		The Engineering Units of tag/pen
Min Value		The minimum possible value of tag/pen
Max value		The maximum possible value of tag/pen.
Selection		Press this button to select a different tag for this pen
Remove		Press this button to completely remove this pen from the legend and the trend chart
Hide		Select (check) this option to hide this pen in the trend chart
Pen style		Press this button to change the line style, weight, color, markers, etc. of the pen
Scale		Select (check) this option to display the pen scale on the trend chart
Description		Description of the tag/pen
Current		The current value of the tag configured for the pen
Cursor		The value of the pen where it intersects the cursor line

- **Properties:** Enables configuration of the properties for the field highlighted in the **Available** or **Visible** list:

Property	Description
Label	Label for the field displayed during runtime
Width	Width for the field (in pixels) during runtime
Align	Alignment of the data displayed in the field
Available during run-time	If this option is enabled, the user can show or hide the field during runtime

- **Maximum size:** Defines the size of the legend in terms of number of lines. For instance, the user might have 8 points being displayed in the trend object, if the maximum size is set to two, the legend will have a scroll bar to allow the user to scroll to the other points.
- **Number of Items:** Number of points (default) displayed on the legend. Users can be enabled to add/remove points during runtime regardless of the value in this field.
- **Selected Item:** In this field, a numeric tag can be configured. The object writes the number of the selected line into this tag. In addition, different rows can be selected by writing their values in this tag.
- **Fonts:** Sets the font for the text displayed in the legend.

Advanced

Accessing the dialog

To access the "Advanced" dialog for a specific trend object, first select the [Object Properties dialog for the screen object on page 281](#) and click on **Advanced**.

Alarms, Events and Trends

The dialog in detail

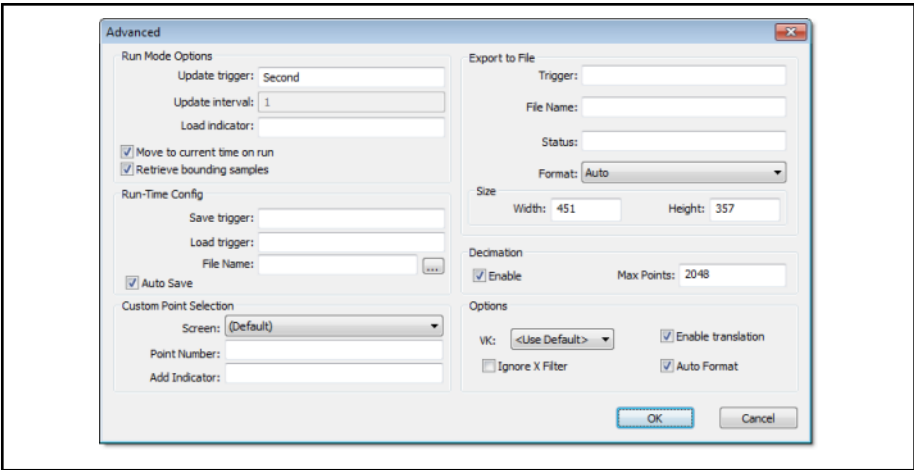


Fig. 9-32: Axes dialog

The "Advanced" dialog contains the following elements:

Pane / Element name		Description
Run mode options	Update trigger	If the tag configured in this field changes value, the trend object is updated (refreshed).
	Update interval	If the update trigger is issued and the X-axis is type numeric, the value on this field is added to the minimum and maximum values of the X-axis.
	Load indicator	Enter the name of a project tag. While the trend object is loading external data, the tag receives a value of 1 and if the trend object has finished loading the data, the tag receives a value of 0.
	Move to current time on run	If this checkbox is enabled, the X-axis automatically shifts to the current time if the object is triggered to Play mode during runtime.
	Retrieve bounding samples	If this box is checked, the object retrieves the output data of the object (first points only). Disabling this option can improve the performance, since the output points of the object are not retrieved from the history. However, the object does not draw lines linking the first and last samples to the maximum limit of the object.

Alarms, Events and Trends

Pane / Element name		Description
Run-time config	Save trigger	The settings of the Trend object modified during runtime can be saved in temporary files. This option can be used as follows: If the tag configured in this field changes its value (e.g., toggles), the current settings of the trend object are saved in the temporary file. This command is not available for the Thin Client. - Keep the settings consistent, to ensure that the settings configured during runtime are not lost in case the screen is closed and reopened or the project is restarted. - Create standard settings for different scenarios and load the appropriate configuration during runtime, based on a pre-defined condition or based on the user-selection.
	Load trigger	If the tag configured in this field changes its value (e.g., toggles), the settings from the temporary file are loaded and applied to the trend object during runtime. Note: After the screen with the configured trend object is saved, the settings are not automatically loaded from the temporary file if the screen is reopened, unless the Load trigger command is executed before the screen is closed.
	File name	If this field is left blank, the temporary file is saved in the project Web sub-folder with the syntax ScreenNameObjectIDTrendObject.stmp (e.g., MyScreen10TrendObject.stmp). The Thin Client station saves/loads the temporary file in the standard "Temp" directory of the operating system (e.g., \Documents and Settings\CurrentUser\Local Settings\Temp). A customized file name can be configured for the temporary file in this field or even a string tag between curly brackets, to enable users changing the name of the configuration file dynamically during runtime by changing the value of this tag. If no path is specified, the file is saved in the project Web sub-folder by default.
	Auto save	If this checkbox is enabled, the current settings of the trend are automatically saved in the temporary file in case the screen with the configured trend is closed during runtime. If the checkbox is disabled, the settings are saved only if the Save trigger command is executed.
Custom point selection	Screen	This interface enables creating custom dialogs to modify or insert pens to the object. Name of the screen that has to be launched if a command is triggered during runtime to modify or insert a new pen to the object.
	Point number	Point number (from the Points dialog on page 282), indicating the point associated to the pen that is inserted or modified during runtime.
	Add indicator	Flag that indicates that an action was triggered to insert a new pen (value 1) instead of modifying a pen that is already visualized (value 0).

Alarms, Events and Trends

Pane / Element name		Description
Export to file	Trigger	If the tag configured in this field changes its value (e.g., toggles), the current state of the trend display is exported to an image file. Toolbar and scroll bar are not included. The legend and time display are included only if the trend display is configured to display them.
	File name	The file path and name of the exported file. If no path is specified, the file is saved in the project directory. If no extension is specified, it is defined by Format . Tip: A project tag can be specified in curly brackets (e.g., {tagname}) to change the file name during runtime.
	Status	The tag configured in this field receives status codes that indicate the success or failure of the export.
	Format	The graphic format of the exported file. If Auto is selected, the format is determined by the extension specified under File name . If Auto is selected but no extension is specified, the default format is BMP.
	Size	The image file is exported at full size by default. However, width and height (in pixels) can be specified.
Decimation	Enable	If this option is selected, the trends in the trend object that are configured to display historical data have their data decimated before the trends are drawn. This means for each trend that the X-axis is divided into a number of intervals (determined by Max points) and all of the data points within each interval are averaged together to be drawn as a single point. This can improve runtime performance in case there is a large amount of historical data to display. Additionally, trends can be made easier to read. Decimation only works if the trend object is in Stop mode (or Historical mode). When selecting this option, please observe that decimation is carried out for all trends that have been configured for display of historical data. To use this function only for a single trend, use the Draw mode in the Options on page 285 .
	Max points	The maximum number of data points used to draw each trend. Default is 2048.
Options	VK (Virtual keyboard)	For this object, the virtual keyboard on page 73 type is used.
	Ignore X filter	If this checkbox is enabled, the X filter is ignored to avoid adding WHERE or querying clause to the data source.
	Enable translation	Enable the external translation for the text displayed by this object.
	Auto format	If this checkbox is enabled, the decimal values in the columns on page 294 "Current", "Cursor", "Max", "Min" and "Scale" are formatted according to the virtual table created by the Set-DecimalPoints on page 604 function. Note: For the Auto format function to work, decimals formatting on the X-axis has to be disabled, i.e., the Decimals in the Axes dialog on page 287 have to be left blank.

9.6.4 Using a Text File as Data Source

The trend display can generate trend charts from any text file where values are organized in columns and lines. The columns should be separated from each other by special characters (usually the comma). Each sample (pair of values representing a point in the graph) is represented by a line. Suppose that the user wants to display a chart with the information in the following table:

Alarms, Events and Trends

X value	Y1 value	Y2 value
0	0	10
1	1	20
2	2	30
3	3	40

One variable represents the X-axis and two variables (Y1 and Y2) represent different lines in the chart. The first step is to convert the data into a text file. If commas are used as separators, the file is displayed as follows:

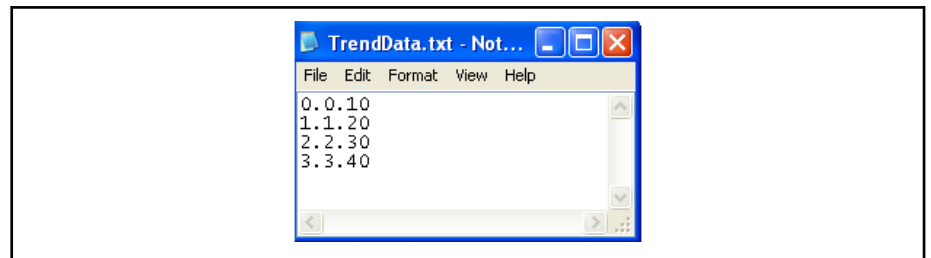


Fig.9-33: Trend data text

It is highly recommend to save the file in the same folder as the project. By doing so, it is not necessary to specify the entire path and the project will still work, even if it is copied to a different computer.

Once the trend object is added to the screen, double-click on the object to open the "Object Properties" and click on **Axis**....

Change the "Data Type" of the "X Axis" to **Numeric** and set the ranges as shown in the picture below:

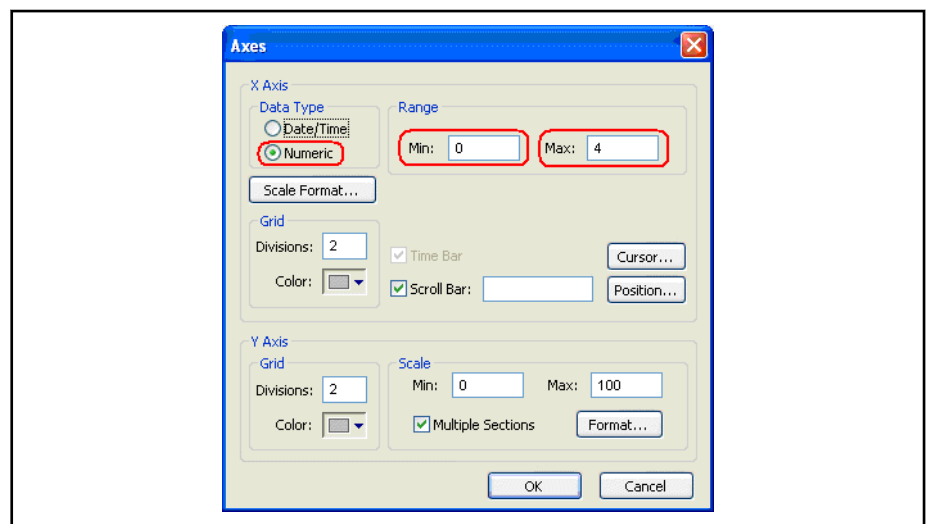


Fig.9-34: Axes dialog

Click on **OK** and, in the "Object Properties" window, on the **Data Sources...** button. The following window is displayed:

Alarms, Events and Trends

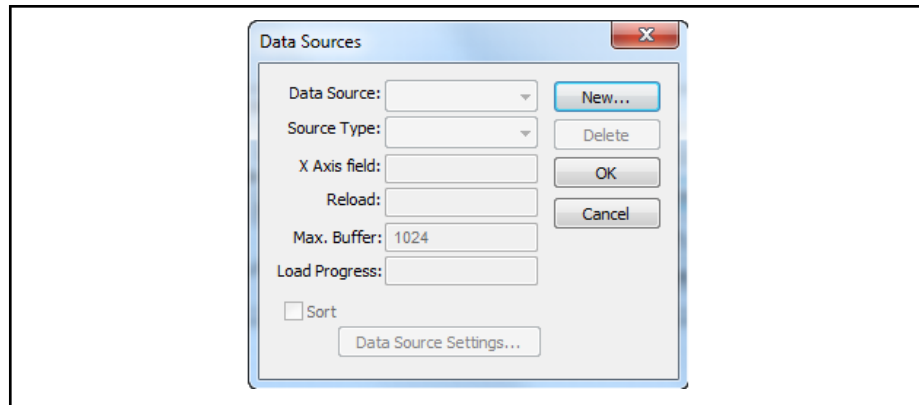


Fig. 9-35: Trend object – Data Sources dialog

A data source has to be created to access the text file. Click on the **New...** button, specify the **Data Source** name **MyTextFile** and click **Create**. Now, the following information is displayed:

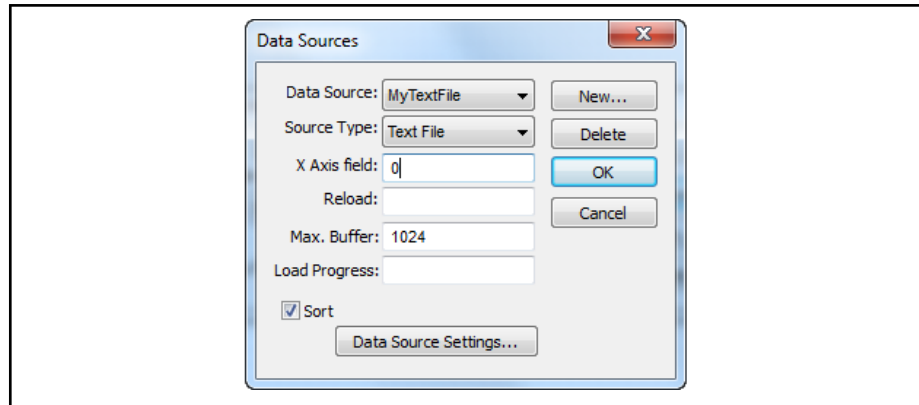


Fig. 9-36: Setting X Axis field to 0

In the **X Axis field**, the column in which the text file represents the X-axis has to be specified. In this example, column 0 is used. Enter "0" for this field and click on **Data Source Settings...**. The following window is displayed:

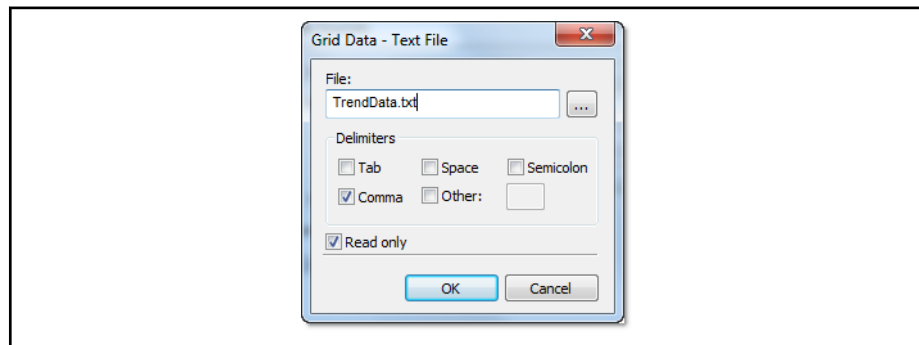


Fig. 9-37: Selecting the text file

If the text file has been copied to the project folder, only the file name has to be specified. Otherwise, enter the complete path of where the file is located (if required, use the browse button). Click on **OK** in this window and on **OK** again to finish the data source configuration and close the "Data Source" configuration window.

Now, Y1 and Y2 have to be defined. These are represented by points on the trend display. Double-click on the trend object to open the "Object Properties" window and click on **Points....**. The next step is to define the points according to the following figure:

Alarms, Events and Trends

After completing these steps, execute the project. The result is to correspond approximately to the following figure:

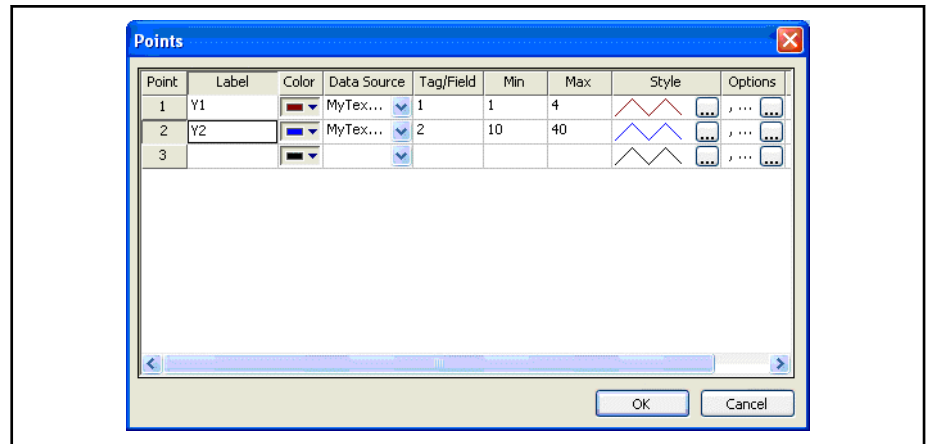


Fig.9-38: Points table

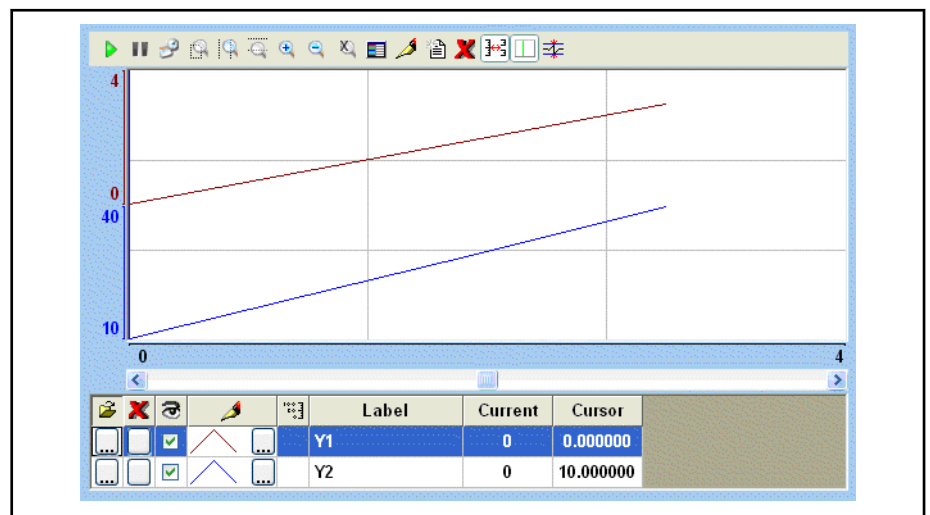


Fig.9-39: Diagram

9.6.5 Using a Database as Data Source

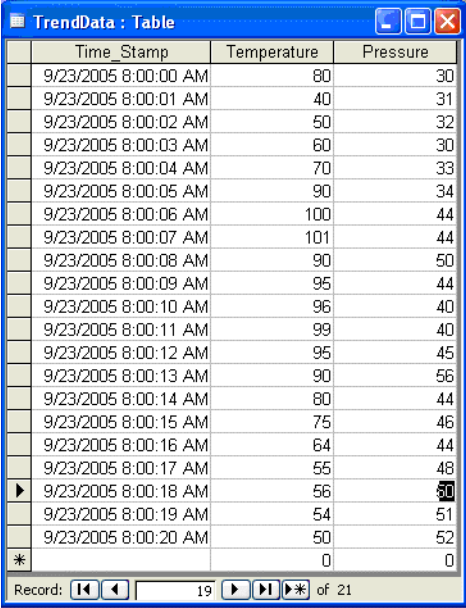
The trend display can generate trend charts from any relational database that can be accessed through the ADO.Net technology.

This appendix describes access to a Microsoft Access database.

If a different type of database is used almost all definitions apply, however, the connection has to be configured differently. For information on how to configure other databases, please refer to the appendixes in the database interface section of this manual.

If an Access database named "mydata.mdb" is stored under C and a graphic is to be created based on the information in the following table:

Alarms, Events and Trends

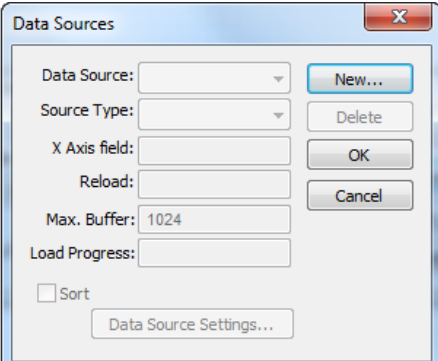


Time_Stamp	Temperature	Pressure
9/23/2005 8:00:00 AM	80	30
9/23/2005 8:00:01 AM	40	31
9/23/2005 8:00:02 AM	50	32
9/23/2005 8:00:03 AM	60	30
9/23/2005 8:00:04 AM	70	33
9/23/2005 8:00:05 AM	90	34
9/23/2005 8:00:06 AM	100	44
9/23/2005 8:00:07 AM	101	44
9/23/2005 8:00:08 AM	90	50
9/23/2005 8:00:09 AM	95	44
9/23/2005 8:00:10 AM	96	40
9/23/2005 8:00:11 AM	99	40
9/23/2005 8:00:12 AM	95	45
9/23/2005 8:00:13 AM	90	56
9/23/2005 8:00:14 AM	80	44
9/23/2005 8:00:15 AM	75	46
9/23/2005 8:00:16 AM	64	44
9/23/2005 8:00:17 AM	55	48
9/23/2005 8:00:18 AM	56	50
9/23/2005 8:00:19 AM	54	51
9/23/2005 8:00:20 AM	50	52
*	0	0

Record: 19 of 21

Fig.9-40: Trend data table

The first step is to add the trend object to the screen. Double-click on the object to open then Object Properties and click on **Data Sources....** The following window is displayed:



Data Sources

Data Source: New...

Source Type: Delete

X Axis field: OK

Reload: Cancel

Max. Buffer: 1024

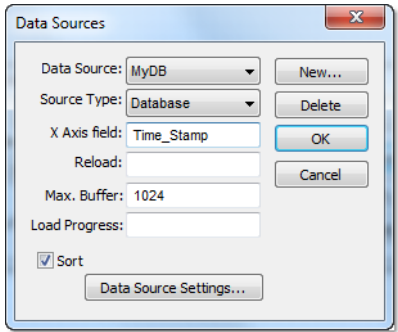
Load Progress:

☐ Sort

Data Source Settings...

Fig.9-41: Trend object – Data Sources dialog

To access the database, a data source has to be created. Click the **New...** button, enter the **Data Source** name **MyDB** and click **Create**. Now, the following information is displayed:



Data Sources

Data Source: MyDB New...

Source Type: Database Delete

X Axis field: Time_Stamp OK

Reload: Cancel

Max. Buffer: 1024

Load Progress:

☒ Sort

Data Source Settings...

Fig.9-42: Setting the X Axis field to Time_Stamp

Alarms, Events and Trends

Change the **Source Type** to **Database** and specify "Time_Stamp" in the **X Axis** field. Click on the **Data Source Settings** button. The following window is displayed:

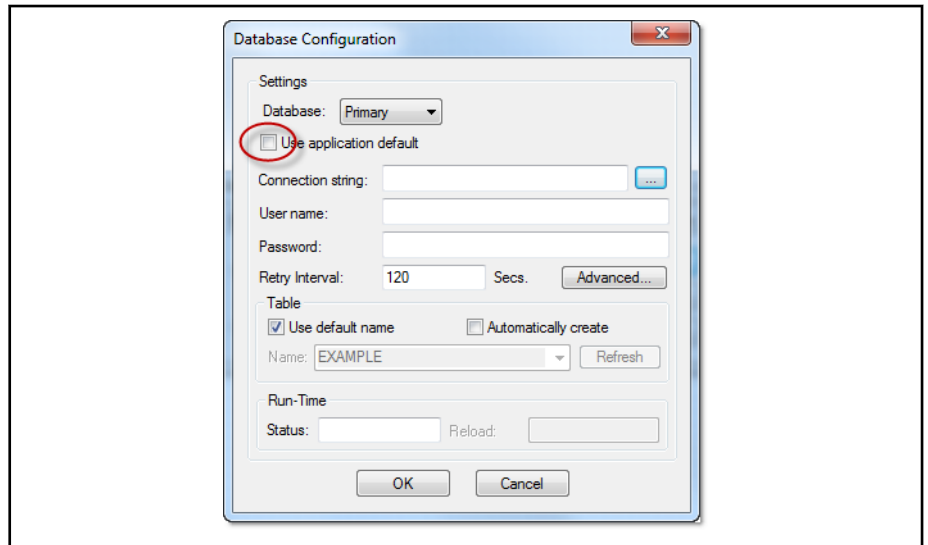


Fig.9-43: Disabling the Use application default option

Disable the checkbox **Use application default** and click on the browse button ... to configure the Connection string. The following window is displayed:

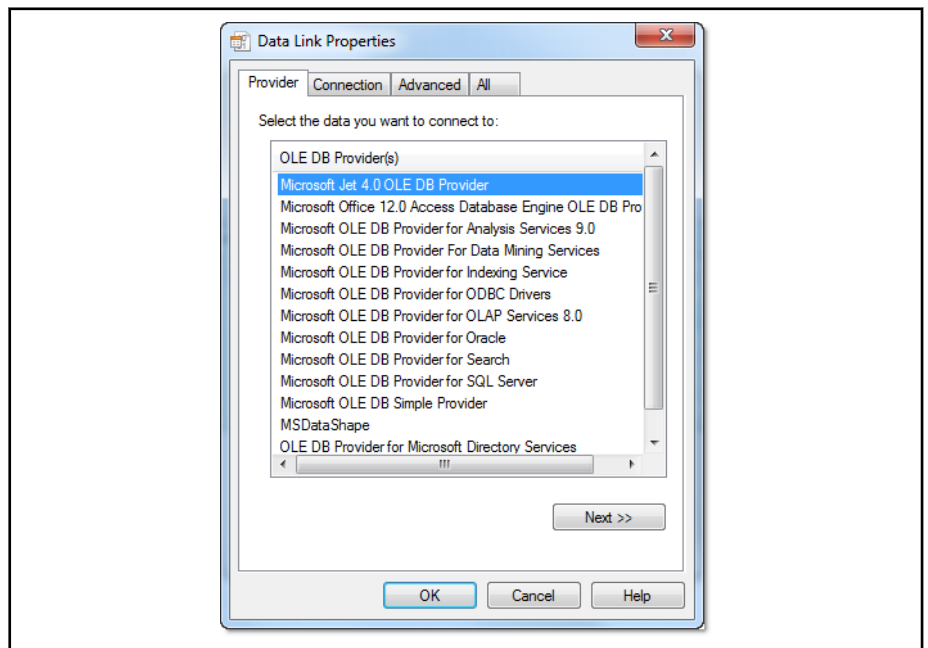


Fig.9-44: Selecting the OLE DB Provider

Select the Microsoft Jet 4.0 OLE DB Provider and click on **Next »**. In the following window, the database path is to be specified:

Alarms, Events and Trends

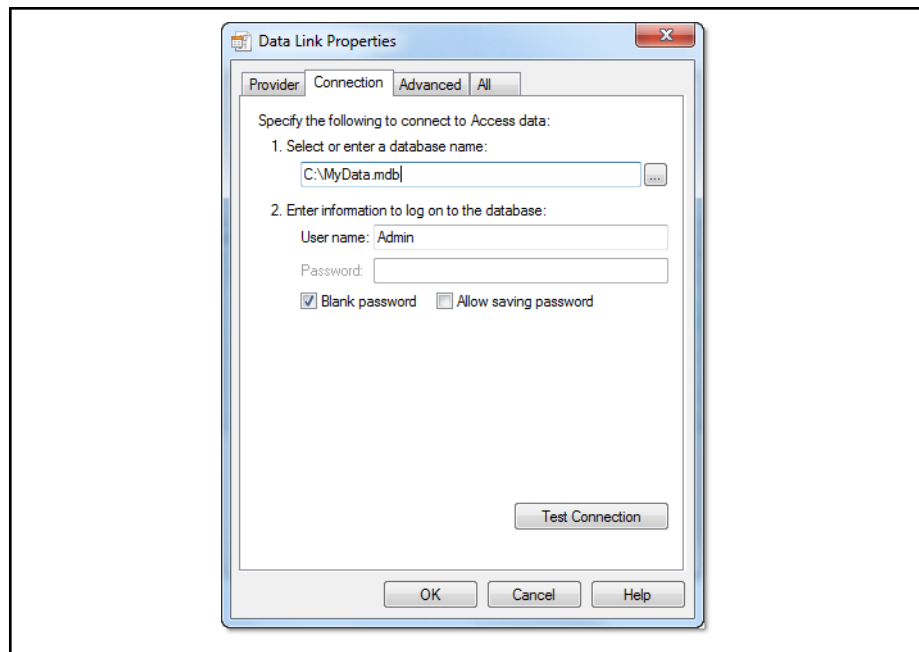


Fig.9-45: Selecting the database file

Click **OK** to finish the Connection string configuration. Disable the option **Use default name** and select the table from the database as shown in the following:

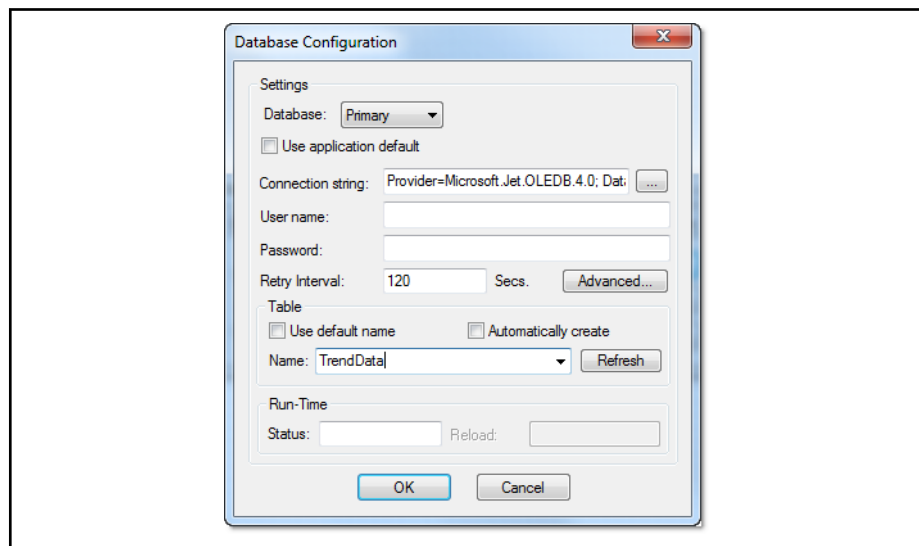


Fig.9-46: Selecting the table in the database

Click on **OK** in this window and on **OK** again to finish the data source configuration and close the "Data Source" configuration window.

Now Temperature and Pressure have to be defined. These are represented by points on the trend display. Double-click on the trend object to open the Object Properties window and click on **Points....** The next step is to define the points according to the following figure:

Alarms, Events and Trends

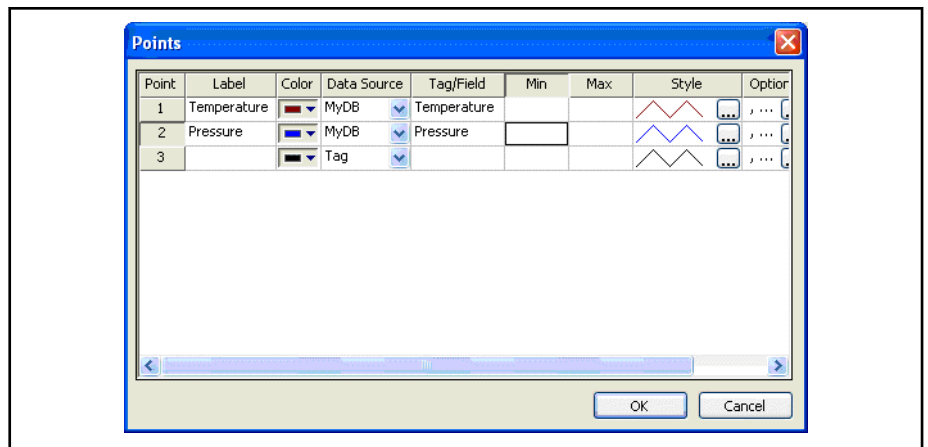


Fig.9-47: Points table

If the trend display is executed, it starts with the current date/time. To display the data in the chart the start date/time has to be properly configured as follows:

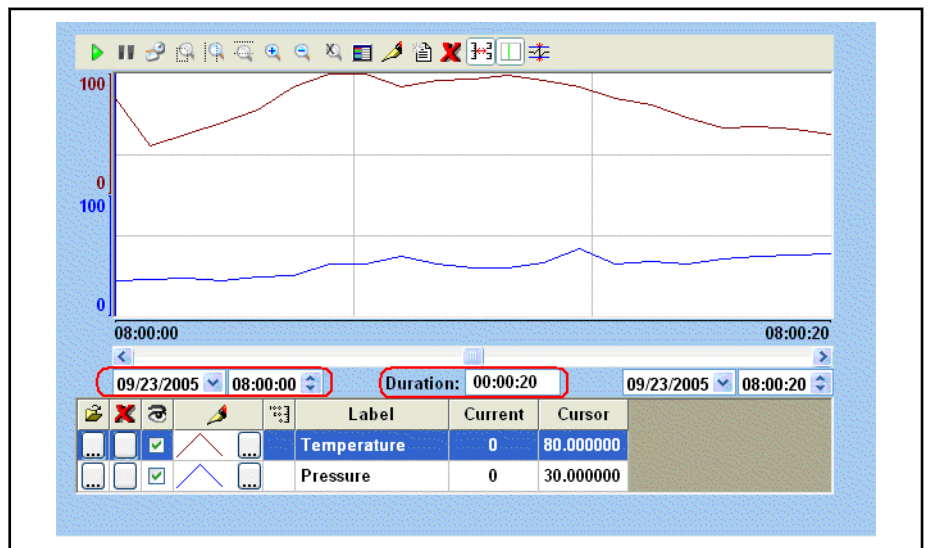


Fig.9-48: Diagram

9.7 Table Object

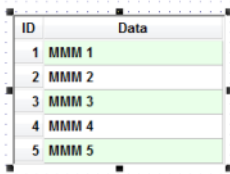
9.7.1 Table Object, General Information

The table object enables reading/writing data in the table format from/to the data source configured in the object.

To draw a table object, proceed as follows:

1. In the "WinStudio Screen Tools" tool window, click on "Table" .
2. Click on the screen, left-click and drag the mouse across the screen to create a box of the desired size (while holding down the mouse button).
3. Release the mouse button to display the table object.

Alarms, Events and Trends



ID	Data
1	MMM 1
2	MMM 2
3	MMM 3
4	MMM 4
5	MMM 5

Fig.9-49: Example of a table object

Right-click on the table object and select **Properties** from the menu. The Object Properties dialog is displayed. Use this dialog to configure the parameters of the table object:

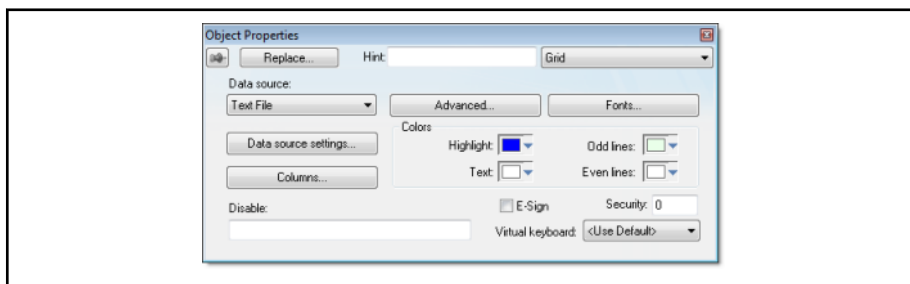


Fig.9-50: Object Properties: Table

- **Data source:** Select the data source type. The object supports three data sources:

Data source	Description
Text file	Displays data from a text file in the ASCII or Unicode format (e.g., CSV text files).
Class tag	Displays values from a class tag, where the class elements of the tag are fields (columns) of the table object and each array position represents one line of the table object.
Database	Displays data from an SQL relational database, using ADO (ActiveX Database Object) to exchange data with the database.

- **E-Sign:** If this option is enabled, the user is prompted to enter the electronic signature before entering or modifying data on the object.
- **VK:** Select the "Virtual keyboard" type used for this object. The option <Use Default> selects the default virtual keyboard configured in the "Viewer" settings (**Viewer** tab in the Properties dialog of the "WinStudio" folder).
For this table object, also a different virtual keyboard can be specified.
- **Security:** Enter the security access level required for the object/animation.
- **Disable:** In this field, an expression can be entered to disable data input or action by the user.
- **Highlight:** Select a background color for the selected line during runtime.
- **Text:** Select a text color for the selected line during runtime.
- **Odd lines:** Select a background color for the odd lines.
- **Even lines:** Select a background color for the even lines.
- **Fonts:** Click to open the "Fonts" dialog to configure the font settings for the text displayed in the table object.
- **Columns:** Click to open the [Columns dialog on page 307](#) to configure the settings (such as label, column, width, etc.) for the columns of the table object.

- **Data source settings:** Click to open the [Data dialog on page 309](#) to specify a data source for the table object.
- **Advanced:** Click to open the [Advanced dialog on page 311](#) to configure various settings for the table object.

9.7.2 Columns Dialog

To configure the settings for each column displayed by the [grid object on page 305](#) during runtime, proceed as follows:

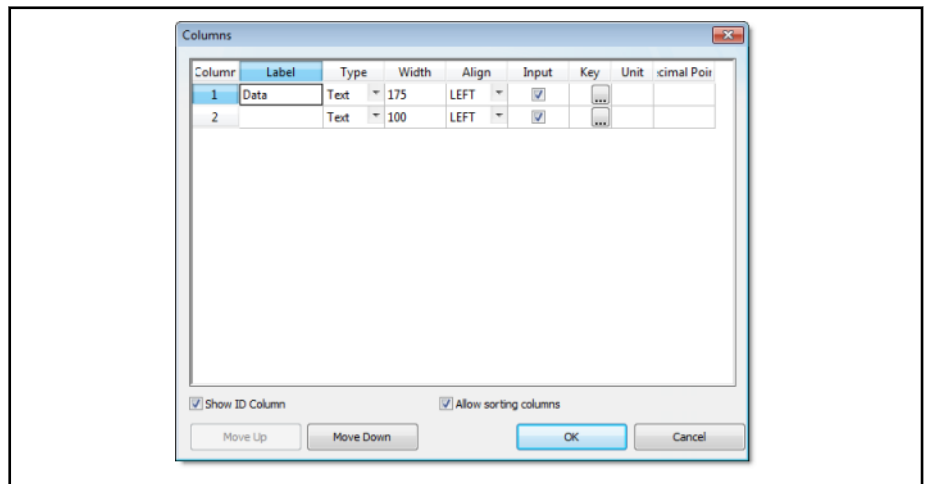


Fig. 9-51: Columns dialog

- **Column:** The ID number defines the position of the column in the table.
- **Label:** Enter a label for each column that is displayed as the heading (first) line of the table object.



In the **Table object** field, tags can be configured in curly brackets to modify the field dynamically during runtime. If the label is left blank (e.g., "") the width of the column is set to 0 during runtime. This option is useful to hide columns during runtime.

- **Field:** Enter the name of the field (column) in the SQL relational database to which the table object is linked. If this field is left blank, the text configured in the **Name** field is used as the Field name. (This setting is only available if the **Data Source** type is set to Database.)
- **Type:** Select the type of interface that is used in the column. The following options are available:

Type	Description
Text	Displays alphanumeric values
Numeric	Displays numeric values
Screen	Displays the picture (*.bmp or *.ico format) of the data source. For instance, if the value of the data source is MyFile.bmp , the table object displays the picture from the file MyFile.bmp in the project folder. The picture is automatically resized to fit the cell of the table object. The picture file(s) has/have to be stored in the Web sub-folder of the project folder to support this feature on Thin Client stations. CView projects support pictures in bitmap format (*.bmp), however, not in icon format (*.ico).
Checkbox	Displays a checkbox. This checkbox is disabled if the value read from the file is 0, <NULL> or "FALSE"; otherwise, the checkbox is enabled. By default, "WinStudio" uses the value 0 for disabled and the value 1 for enabled.

Alarms, Events and Trends

Type	Description
Time	Displays the value in the time format (e.g., HH:MM:SS). This setting is only available if the Data Source type is set to Database .
Date	Displays the value in the date format (e.g., MM/DD/YYYY). This setting is only available if the Data Source type is set to Database .
Date/Time	Displays the value in the date/time format (e.g., MM/DD/YYYY HH:MM:SS). This setting is only available if the Data Source type is set to Database .



- If the **Data Source** type is set to Database, it has to be ensured that the **Type** for each column configured in the object matches the type of the respective field in the database.
- If the **Data Source** type is set to Database, valid SQL statements can be configured directly in the field (e.g.,
List(DISTINCT [Cell_Name]) AS [Cell_Name].
 Tag names can also be configured in curly brackets to modify this setting during runtime (e.g., **{MyFieldName}**).



If the column type is set to Picture, the grid object displays a default icon () in case the picture file is not found during runtime. To configure a different picture to be displayed when the file is not found, copy the picture file to the "Web" sub-folder of the project folder and configure its name in the "project_name.app" file as follows:

[Objects]

GridDefaultPicture=PictureFileName

- **Width:** Enter a column width in pixels
- **Align:** Select an alignment for the data shown in the column. There are three options: Left, Right or Center
- **Input:** Clicking on (check) the checkbox enables users to enter data in this column during runtime
- **Key:** Use this field to designate a shortcut for sorting the values. A shortcut is a combination of keys simultaneously pressed on a keyboard (e.g., <CTRL>+<C>, <CTRL>+<V>, etc.). This option is especially useful when creating projects for runtime devices that do not provide a mouse or touchscreen interface and only have a keyboard to access the project during runtime.
- **Unit:** Enter the name of the engineering unit (i.e., the unit of measurement), if any, that applies to the data displayed. String tags can be entered using the **{tagname}** syntax to change the **Unit** value during runtime.
- **Decimal Points:** Enter the number of decimal points to be displayed. Integer tags can also be entered using the **{tagname}** syntax to change **Decimal Points** value during runtime.



If the **Data Source** type is set to Class Tag and the Columns dialog is left blank, the object displays the values from all members of the class tag with the following default column settings:

- Label = <Name of the member from the class tag>
- Type = Text
- Width = <Minimum size to display the name of the member from the class tag on the header of the grid object>
- Align = Center
- Input = Enabled (checked)
- Key = <None>
- Unit = <Unit of the class tag element>

The unit of a class member or tag is set using the [Tag Properties on page 124](#) tool.

- **Show column ID:** Check to automatically display the number of each line.
- **Allow sorting columns:** Check to enable the user to sort the values in the columns during runtime, either by clicking on the label or by using the shortcut configured for each column. This option is disabled if the **Show header** option in the [Advanced dialog on page 311](#) is disabled.



Use the **Move Up** and **Move Down** buttons to reorder the display of the columns.

9.7.3 Data Dialog

Grid data – Text file

This dialog enables configuring the data source for table objects.

If the **Data Source** type is set to Text File, the following settings are available for configuration:

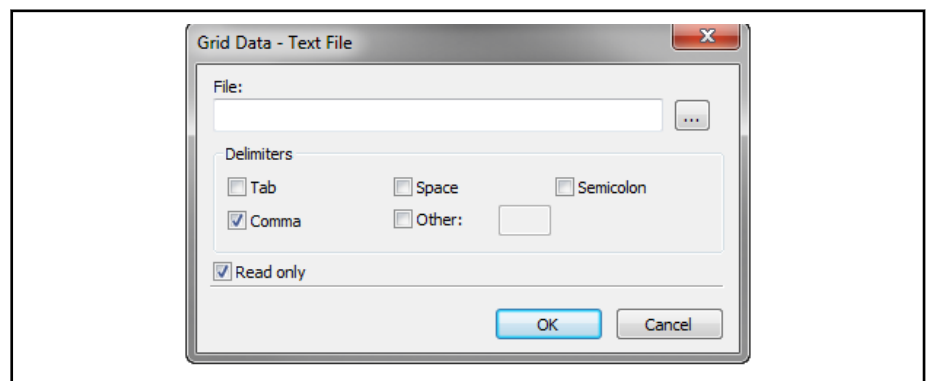


Fig. 9-52: Grid Data - Text File

- **File:** Enter the name of the source text file. Enter the file name and path or click the ... button to browse for it. (If the file is stored in the project folder, the path does not need to be specified in the name.)



In the **File** field, tag names can be configured between curly brackets (e.g., {TagName})

Alarms, Events and Trends

- **Delimiters:** Set the delimiter(s) used in the data source file. For instance, if the data is to be read from a CSV (comma separated values) file, select the **Comma** option. A custom delimiter can be selected by checking the **Other** option and typing it in the field beside it.
- **Read only** checkbox: If this option is enabled, the table object only reads data from the specified file. The object does not write anything to the file.

Grid data – Class tag

If the **Data Source** type is set to Database, the following settings are available for configuration:

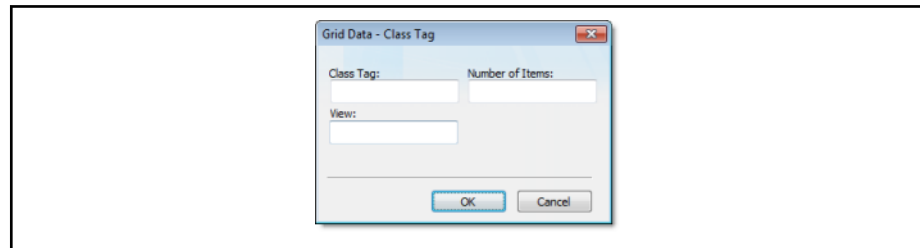


Fig.9-53: Grid Data – Class Tag

- **Class Tag:** Enter the name of the main class tag source. (Do not specify a specific member of the class tag.) The initial array position can be specified in this field (e.g., "Mytag[10]"); otherwise, 0 (zero) is used as initial position by default.
- **Number of Items:** Enter the number of Array positions of the **Class Tag** to be displayed.
- **View:** If the tag configured in the optional field changes its value (e.g., toggles) during runtime, the table object opens a dialog to display/hide each column or modify their positions.

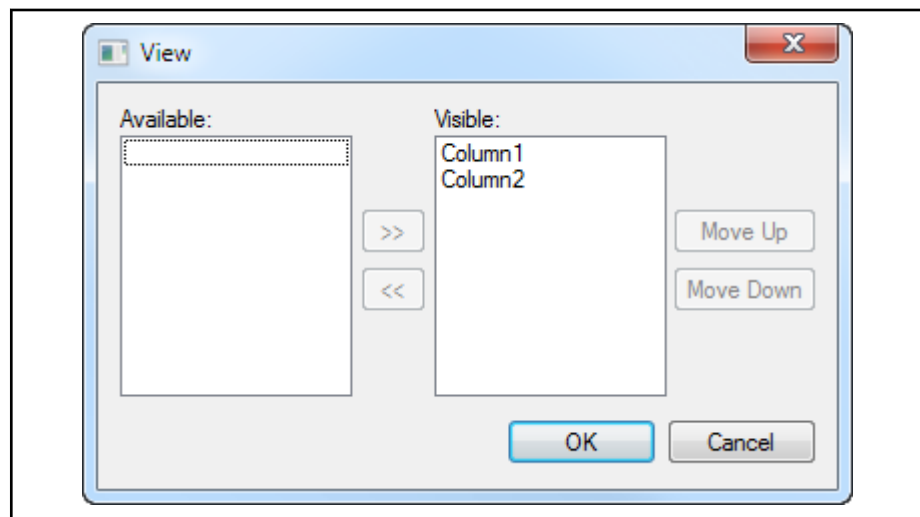


Fig.9-54: Dialog: View

Grid data – Default database configuration

If the **Data Source** type is set to Database, the following settings are available for configuration:

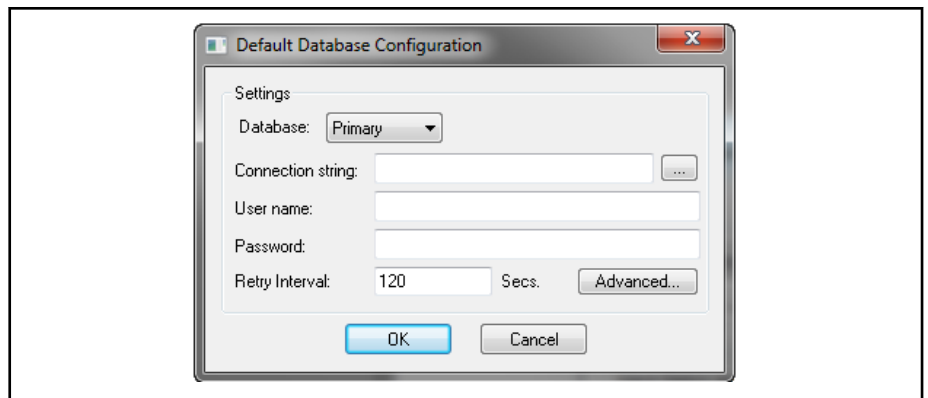


Fig.9-55: Grid Data - Default Database Configuration

For further information about this dialog, refer to the [Database Configuration dialog on page 438](#).

9.7.4 Advanced Dialog

In this dialog, the advanced settings for table objects can be configured.

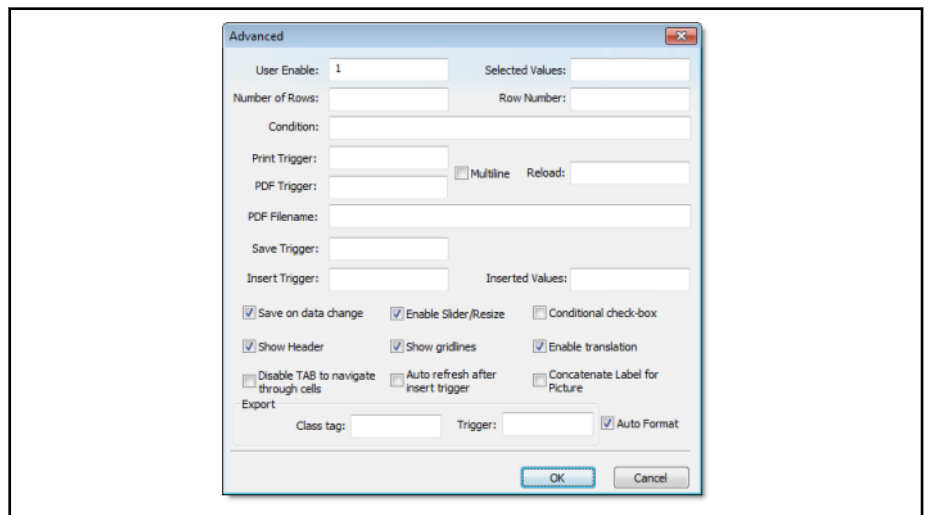


Fig.9-56: Advanced dialog

- **User Enable:** If the value of this tag is TRUE (different from 0), the user can select different lines of the object by clicking on them during runtime. This field can be configured with a tag or with a numeric value.
- **Selected Values:** The values from each column of the selected line are written to each position of the array tag configured in this field. Additionally, the value of the cells currently selected in the table object can be modified by changing the value of the array tag configured in this field. In this field, the initial position (offset) can be configured.
- **Number of Rows:** The table object writes the number of rows currently available in the table object to the tag configured in this field.
- **Row Number:** The table object writes the number of the row currently selected during runtime. In addition, different rows can be selected by writing their values in this tag.
- **Condition:** Enter an expression to filter the table data. Only rows that match the expression will be displayed. The expression has to comply with the following syntax:

Alarms, Events and Trends

Column Operator Value

For example...

ColumnX > 200

If **Data Source** (in the [Table Object Properties dialog on page 305](#)) is set to Text File or Class Tag, the "Column" contains the value specified in the Label. If **Data Source** is set to Database, the column contains the value specified in the Field. (In this case, if the Field is left blank, the column value specified is the Label.)

Database expressions have to be formatted like SQL Where statements. The following table illustrates the operators to be used:

Condition Expression Operators

Comparison	Data source is text file...	Data source is database...
equal to	=	LIKE
not equal to	<>	NOT LIKE
Placeholder, single character	?	_
Placeholder, unlimited characters	*	%

As such, the following expression for text file...

C1 = 'ab?d'

... means the same as the following expression for database...

C1 LIKE 'ab_d'

Finally, several expressions can be combined simultaneously in the **Condition** field using the logic operators "AND", "OR" and "NOT".

ColumnAge > '10' OR ColumnName = 'John' AND ColumnDate > '05/20/2003'



Tags can be configured between curly brackets **{TagName}** in the **Condition** field to change the filtering condition during runtime.

- **Print Trigger:** If the tag configured in this field is toggled, the current state of the table object is sent to the default printer.
- **PDF Trigger field:** If the tag configured in this field is toggled, the current state of the table object is saved as a PDF file at the location specified under **PDF Filename**.
- **PDF Filename field:** Enter a complete file path and name where the PDF file is to be saved. Tag names using the **{tag}** syntax can also be entered.



PDF Trigger and **PDF Filename** are not supported in projects running on Windows Embedded or Thin Client systems.

- **Multiline** checkbox: If this option is enabled, the print output or PDF is formatted according to the available column space, and the text within each cell is wrapped so that it is completely displayed.

Alarms, Events and Trends

- **Reload:** If the tag configured in this field is toggled, the object reloads the data from the data source and displays it.
- **Save Trigger:** If the tag configured in this field is toggled, the data source (text file or database) is updated with the current values of the grid object. This field is not available if the Data Source type is set to Class Tag, as the values are automatically updated in the tags as cells are changed in the table.
- **Insert Trigger:** If **Auto refresh after insert trigger** is enabled (checked), the tag configured in this field is used as a trigger to refresh the data-base table. If the value of the tag changes, a new row is added to the table and the values of the array configured in the **Inserted Values** field are automatically inserted.
- **Inserted Values:** If the **Insert Trigger** is used, the array tag configured in this field provides the values to be inserted. This field may only contain one array tag, however, it can be of any size.
- **Save on data change:** If this option is enabled, the values are updated on the data source (text file or database) as soon as the user enters a new value into the table during runtime. This option is disabled if the Data Source type is set to Class Tag, as the values are automatically updated in the tags as the cell values are changed in the table.
- **Enable Slider/Resize:** If this box is disabled, it is neither possible to scroll the list by dragging the slider button nor to change the cell size during runtime.
- **Conditional check-box:** If this option is enabled, checkboxes on the Grid cannot be disabled during runtime, unless all preceding checkboxes in the same column are already disabled. This option is particularly useful if users are required to follow a pre-defined sequence. This field is not available if the Data Source type is set to Class Tag.
- **Show Header:** If this option is enabled the header of the table object is visible during runtime, displaying the label of each column.
- **Show gridlines:** If this option is enabled, the gridlines of the table object are visible during runtime.
- **Enable translation:** If this option is enabled, the text displayed by the grid object is provided to the Translation tool during runtime.
- **Disable TAB to navigate through cells:** If this option is enabled, it is possible to navigate the cells of the table object with the arrow keys instead of the Tab key. The Tab key is to be disabled for navigation if it is used for switching to the next screen object that can be activated.
- **Concatenate Label for Picture:** If this option is enabled, the reference name for the picture is the result of the concatenation of the name in the Field column with the value of the Label column. The result is <Label name>_<Field value>.
- **Export:** This interface enables exporting data from the table object to a class array tag, regardless of the data source selected for the object. The following fields have to be configured to support this function:

Alarms, Events and Trends

Field	Description
Class tag	Enter the main tag name of the class array tag that is to receive the exported values. Each row in the grid object is exported to one array position of the array tag by the respective column labels. The initial array position can be configured in this field; 0 is the default.
Trigger	If the tag configured in this field changes its value (e.g., toggles), the data is exported from the table object to the class array tag configured in the Class Tag field.



The Export feature is an uncomplicated and powerful tool to transfer data from different data sources to tags. After exporting the data to tags, different tasks can be used to handle the data, such as the [FileWrite on page 583](#) function or the Recipe or Report tasks to save data in text files (e.g., CSV files).

- **Auto Format:** If this option is enabled, decimal values in **Numeric** columns are formatted according to the virtual table created by the [SetDecimalPoints on page 604](#) function. This option only works in columns in which no **Decimal Points** have been configured yet. For further information, refer to [Grid object: Columns dialog on page 307](#).

10 Background Tasks

10.1 Background Tasks, General Information

Background tasks are, as the name implies, project features that run in the background, as opposed to the graphical screens for user interaction.

Background tasks are executed by the Background tasks module (refer to <Execution tasks>) and they are defined by task worksheets in the Project Explorer.

10.2 Alarm Worksheet, General Information

The Alarms folder enables the configuration of alarm groups and tags related to each group. The Alarm worksheet defines the alarm messages generated by the application. The primary purpose of an alarm is to inform the operator of any problems or abnormal condition during the process so he can take corrective action(s).

The Alarm worksheet is executed by the Background Task module (refer to <Execution Tasks>). It manages the state of all alarms and saves the alarm messages to the history if configured to do so. The Alarm worksheet itself, however, does not display alarm messages at runtime. To display alarms, the [Screen object Alarm/Event Control on page 262](#) in the "WinStudio Screen Tools" tool window has to be inserted into a WinStudio screen and configured.

To create a new worksheet, open the context menu of the "Tasks" folder and select **Add ► Alarm**.

To edit an existing alarm worksheet, double-click the element in the Project Explorer.

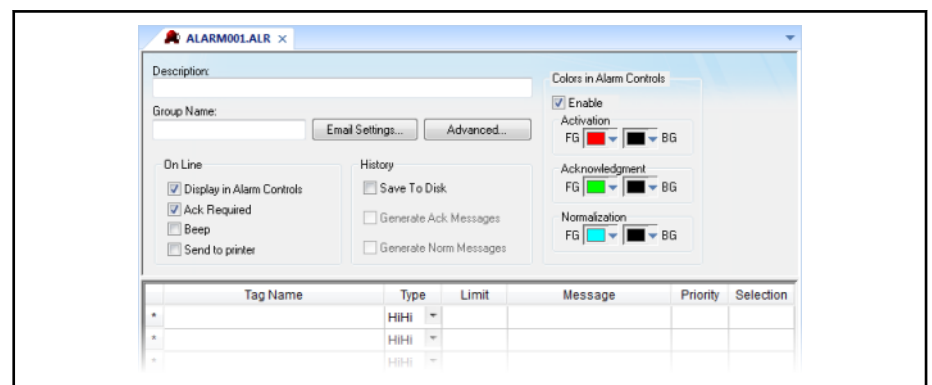


Fig. 10-1: Alarm worksheet

Multiple alarm groups (worksheets) can be created and each group can be configured with independent settings, such as message colors, history log enabled/disabled, etc.

Each alarm worksheet is composed of two areas:

- **Header:** Settings applied to all tags and alarms configured in the same alarm group. These settings enable configuration of the formatting of the messages and actions that have to be triggered based on alarm events (e.g., print alarms, send alarms by email, etc.). For further information, refer to [header settings on page 248](#).
- **Body:** Selecting tags and defining conditions on which alarm messages are created. For further information, refer to [body settings on page 253](#).

Background Tasks



- Alarm groups can be configured to send notifications by Email automatically, based on alarm events. For further information, refer to [email settings on page 250](#).
- The alarm properties associated with each tag (configured in the body of the alarm group) can also be edited on the [Tag Properties on page 140](#) dialog (**Properties** in the "Cross references" tool window). However, before associating a tag with an alarm group, it is necessary to create the alarm group and to configure the settings on its header. These will be applied to all tags associated with the group.
- As of "WinStudio" v6.1+SP2, the Alarm task has been modified to avoid automatically acknowledging alarms by another alarm. For example, the Hi (Lo) alarm is not to be automatically acknowledged when the HiHi (LoLo) alarm becomes active. To enable this behavior, set the following key in the project file (.APP):

[Alarm]

UseLegacyPriorityAck=1

NOTICE

The settings configured in the body of each alarm worksheet are stored in the tags database archive(s). Therefore, changes to the tags database may affect the content of the alarm worksheets (body).

Please note that each tag/type cannot be available in more than one alarm group simultaneously as the alarm group is a property associated to each tag/alarm type

(e.g., Tag: Level; Alarm Type: Hi; Alarm Group: 2).

10.3 Trend Worksheet, General Information

The "Trend" folder enables configuring history groups that store trend curves. The trend worksheet can be used to specify tags with values that have to be stored on disk and to create history files for trend graphs. The project stores the samples in a binary history file (*.hst) and shows both history and online samples in a trend graph screen.

The trend worksheet is executed by the Background Task module. It manages saving of trend data in the history, however, does not display this data to the operator. To display trend data, the [trend object on page 281](#) in the "WinStudio Screen Tools" tool window has to be inserted into a WinStudio screen and configured.

To create a new trend worksheet, open the context menu of the "Tasks" folder and select "Trend" in the "Add" menu.

To edit an existing trend worksheet, double-click it in the Project Explorer.

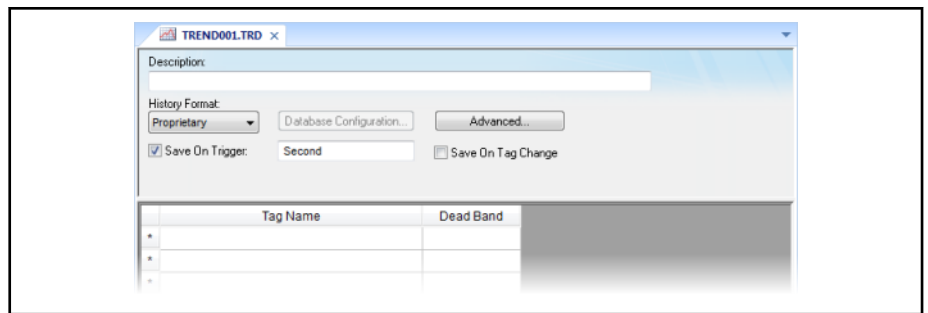


Fig. 10-2: Trend worksheet

The trend worksheet is divided into two areas:

- Header area (top section) containing information for the whole group
- Body area (bottom section) to define each tag in the group. This section contains several columns (only two are shown in the preceding figure).

Use the header parameters on this worksheet as follows:

- **Description** field: Enter a description of the worksheet for documentation purposes.
- **History Format**: Click to select a trend history format. The following options are available:
 - **Proprietary**
 - **File Format**: Binary
 - **Default path**: ...\\project_name\\Hst\\GGYYDDMM.HST, where:
 - YY = two last digits of the year
 - MM = month
 - DD = day



"WinStudio" provides the **HST2TXT.EXE** and **TXT2HST.EXE** programs to convert trend history files from binary (**.hst**) to text (**.txt**) and vice versa. For further information on these programs, refer to [Converting trend history files from binary to text on page 272](#) and [Converting trend history files from text to binary on page 273](#).

- **Database**
 - **Database type**: Selected by the user
 - **Default table name**: TRENDGGG (GGG = trend worksheet number; e.g., TREND001 for the trend worksheet 001)

For further information about the structure of the database table that "WinStudio" uses to save history files, refer to [Database interface on page 429](#).

Background Tasks

NOTICE

String tags can be specified in many fields of the trend worksheet to change these values at runtime. However, this may influence how these values are stored in the trend history:

- If the history format is **Proprietary**, the value of the String tag is converted to a numerical value (if possible) and then saved to the history file. If numeric conversion is not possible, a value of 0 is saved.
- If the history format is **Database**, the actual value of the String tag is saved in the database.

- **Database Configuration:** Opens the [Database Configuration on page 438](#) dialog, to enter the required settings to link the project to an external SQL relational database for the purpose of saving the trend history.
- **Save On Trigger** checkbox and field: Click on (enable) the checkbox and type a tag name to save trend samples if someone changes the specified tag. (The tag change can be an event from the Scheduler.)
- **Save On Tag Change** checkbox: Click (enable) to always save the trend sample in case of a value change in any of the tags from that group.
- **Advanced:** Click to display the "Trend Advanced Settings" dialog. For information about completing the fields in this window, refer to [Batch history configuration on page 273](#).

Use the body parameters on this worksheet as follows:

- **Tag Name** field: Enter the variable name to be saved in the history file.
- **Dead Band** checkbox: Enter a value to filter acceptable changes when **Save On Tag Change** is used.

For example:

The dead band has a value of 5. If the tag value is set to 50 and changes to 52, the system does not register this change in the database as it is smaller than 5. If the change is at least 5, the new value is saved in the history file.

- **Field:** Name of the field in the database where the tag is stored. If this field is left blank, the name of the tag is used as the tag name. For array tags and classes the characters "[", "]" and "." are replaced by "_".

Examples:

Tag name	Default field
MyArray[1]	MyArray_1
MyClass.Member1	MyClass_Member1
MyClass[3].Member2	MyClass_3_Member2



The Trend task can accept only up to 240 tags in a single worksheet. If more than 240 tags are manually configured in the same worksheet, the Trend task generates an error on running the finished project.

Also refer to:

- [Converting trend history files from binary to text on page 272](#)

- [Converting trend history files from text to binary on page 273](#)
- [Creating batch history on page 273](#)
- [Configuring a default database for all task histories on page 442.](#)

10.4 Recipe Worksheet

Recipe worksheets are used to load tag values from or save tag values to an external file during runtime. It is typically used to execute process recipes, however, any type of information (such as operation logs, passwords, etc.) can be stored in the external file.

To create new recipe worksheets, open the context menu of the "Tasks" folder and select **Add... ► Recipe**

To edit an existing recipe worksheet, double-click on it in the Project Explorer.

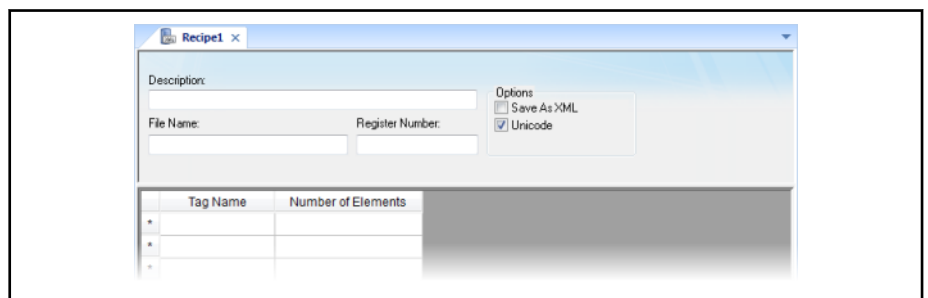


Fig. 10-3: Recipe worksheet

The worksheet is divided into two areas:

- "Header" area (top section) containing information for the whole group
- "Body" area (bottom section) to define each tag in the group.

To use the "header" parameters on this worksheet, proceed as follows:

- **Description:** Enter a description of the worksheet for documentation purposes.
- **File Name:** Enter the name of the external file, using static text (**File1**) or indirect tag (**{FileNameTag}**).
- **Register Number:** Enter a tag to define the register number to be read or written into a DBF file. (This field is for legacy systems only and is no longer used.)
- **Save As XML:** Enable (check) to save information in XML format or disable (uncheck) to save in DAT format. Use a second recipe worksheet to load information from a **.DAT** file to different tags. In contrast, information in **.XML** files have to be loaded to tags with the same name as the tag containing the original data.



Similar to HTML pages, the web server has to be executed to view XML data from the web. Unlike the HTML pages in the runtime system, XML pages do not require the project to be running to view the XML data. (To view XML data, Internet Explorer version 5.0 or higher is required.)

- **Unicode:** Click (enable) to save the recipe in Unicode format (two bytes per character) or (disable) to save the recipe in ANSI format (one byte per character).

Background Tasks



Worksheets can be saved under any name (a predefined file name is not required). A configuration file using the default extension **.RCP** (or **.XSL** if **Save As XML** is selected) contains the recipe configuration and the **File Name** field contains the data file name to be read or written.

To use the "body" parameters on this worksheet, proceed as follows:

- **Tag Name:** Enter tag names to be updated with file contents or with values that are to be written to a file. If the tag is an array, the first position to be used has to be specified. If the tag is an array or a class (or both), all array positions and class elements are automatically activated by default. To configure a specific array position and/or class element, enter it in the **Tag Name** field as usual. For example, `level[3].member`.
- **Number of Elements** field: Specify how many positions of the array tag are in use.



In this field, tag names can be configured in curly brackets (**{tag-name}**) to dynamically change the number of elements configured in the recipe for each array tag during runtime.

To execute a recipe worksheet, the [Recipe on page 558](#) function can be used anywhere an expression is allowed.

10.5 Report Worksheet

Report worksheets are used to design reports that are dynamically generated during runtime (using the current values of the included tags) and then either sent to a printer or saved to a file.

To create new report worksheets, open the context menu of the "Tasks" folder and select **Add... ► Report**

To edit an existing report worksheet, double-click on it in the Project Explorer.

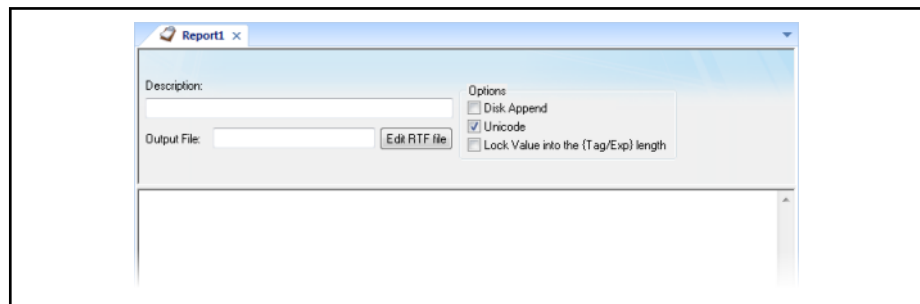


Fig. 10-4: Report worksheet

"Report" worksheets are divided into two areas:

- "Header" area (top section) containing information for the whole group; and
- "Body" area (bottom section) to define each tag in the group.

Use the header parameters on this worksheet as follows:

- **Description:** Enter a description of the worksheet for documentation purposes.
- **Output File:** Enter a tag name for the output file (using the **{tag}** syntax) where data is stored during output for printing to a file. The tag value is part of the file name.

For example: **report{Day}.out**. The generated file might be **report1.out**, **report2.out**, **report3.out**, etc., according to the tag day value.



Report configuration files use ".RCP" as default extension. The data is saved to the file specified in the **Output File** field.

- **Edit RTF file** button: Click to access the report as an RTF file for editing the layout, etc.
- **Disk Append** checkbox: When printing to a file
 - Enable the checkbox to add (change) the new report at the end of an existing file
 - Disable the checkbox to replace the existing report in that file by the new report
- **Unicode** checkbox: Click (enable) to save the report in Unicode format (two bytes per character) or (disable) to save the report in ASCII format (one byte per character).
- **Lock Value into the {Tag/Exp} length** checkbox: Click (enable) to automatically limit the values of tags/expressions in the report to fit between the curly brackets, as they are positioned in the body of the report (see below). This helps to preserve the layout of the report. If this option is left unchecked, the full values of tags/expressions in the report are displayed.

Use the "body" of this worksheet for report formatting. Reports can be configured using data in the system and indicating where to print the tag values. Each tag name replaces the "{tag_name}" tag name. For Real type tags, use the following syntax: **{tag_name n}**, where **n** is the number of decimal places to be printed.



If the standard report editor (text only: ASCII or Unicode) is used, the number of characters reserved for the tag value corresponds to the number of characters used to type the tag name (including the two "curly" brackets). For example, if **{TagA}** is configured in the report body, six characters are to be reserved for the tag value in the report file. This functionality does not apply to reports in RTF format.

To execute a report worksheet, the [Report on page 558](#) function can be used anywhere an expression is allowed.

10.6 ODBC Worksheet

The ODBC interface runs in a network environment and uses the default Windows ODBC configuration. The "ODBC" task can be used for data exchange between "WinStudio" and any database supporting this interface.



In addition to configuring the "ODBC" worksheet, the Windows ODBC default driver has to be configured. "WinStudio" refers to the User DSN (Data Source Name) configured by means of the Control Panel. For further information, refer to the Windows documentation.

Additionally, the ODBC interface is not available for projects running on Windows Embedded target systems.

Background Tasks

To create new ODBC worksheets, open the context menu of the "Tasks" folder and select **Add... ► ODBC worksheet**

To edit an existing ODBC worksheet, double-click on it in the Project Explorer.

A new worksheet is displayed as follows.

Fig. 10-5: ODBC worksheet

ODBC worksheets are executed under the "ODBC Runtime" task. However, creating a new worksheet does not automatically activate the task. The task has to be activated in the task configuration (in the WinStudio context menu under **Tools ► Status**) to start during runtime. For further information, refer to <Task Execution>.

"ODBC" worksheets are divided into two areas:

- "Header" area (top section) containing information for the whole group, defining tags to start read and write events, setting return values, handling database access parameters, etc.; and
- "Body" area (bottom section) to define each tag in the group and relate tags to fields in the current tab from the database table.

To use the "header" parameters on this worksheet, proceed as follows:

- **Description:** Enter a description of the worksheet for documentation purposes.
- **Data Source Name:** Enter the same Data Source Name (DNS) specified in the Windows Control Panel containing information about specific database access.
- **User:** Enter a user name to access the database.
- **Password:** Enter the user password.
- **Table:** Enter a table name in the database.
- **Condition:** Enter a search condition or filter.
- **Status:** Enter a tag in which the status value is to be written on ODBC access. The tag is to report 0 for success and use another value for an error code.
- **Transaction Completed:** Enter a tag that changes its value when the transaction is executed.
- **Select Trigger, Next Trigger, Insert Trigger, Delete Trigger or Update Trigger:** Enter a tag to work as a trigger, where each value change causes WinStudio to execute the respective command. At least one trigger field is required.

To use the "body" parameters on this worksheet, proceed as follows:

- **Tag Name:** Enter the names of tags to update with file contents or tags whose values are to be written to a file.

Background Tasks

- **Column:** Enter the data location in the file (for example, "R3CH" corresponds to Row 3, Column H of an Excel sheet)

Use the Windows Control Panel to set up the ODBC interface for Excel files. Proceed as follows:

1. Click on **Start ► Settings ► Control Panel**.
2. In the "Control Panel" window, double-click on the ODBC icon to open the "ODBC Data Source Administrator" dialog.
3. In the "ODBC Data Source Administrator" dialog, click on Excel Files in the User Data Sources list and on the Configure button.
4. In the "ODBC Microsoft Excel Setup" dialog, enter the Windows configuration name to be used in the DSN field on the "ODBC" worksheet into the Data Source Name field.
5. Click the Select Workbook button to configure the Excel file to be used.
6. Return to the ODBC Data Source Administrator dialog and check whether the User DSN is displayed in the list. Click OK to close the dialog.
7. After configuring the ODBC Windows interface, the "ODBC" worksheets of the project have to be configured.
8. From the "Tasks" tab, insert a new ODBC worksheet.
9. Ensure that the automatic start of the ODBC runtime is specified in the Task configuration (Menu entry **Tools ► Status** in the WinStudio context menu).

To start this configuration, the project simply needs to be executed. The project handles the **Select**, **Next**, **Insert**, **Delete**, and **Update** triggers to enable data exchange in rows in Excel and tags configured in the worksheet.

The meaning of specific error codes can be found in the Windows documentation.

The following is a list of "WinStudio" error codes:

- *Select command*
 - 1 - Error in the ODBCPREPARE function
 - 2 - Error in the ODBCBINDCOL function
 - 3 - Error in the ODBCEXECUTE function
 - 4 - Error in the ODBCSETCH function
- *Next command*
 - 5 - Error in the ODBCSETCH function
- *Insert command*
 - 6 - Error in the ODBCPREPARE function
 - 7 - Error in the ODBCEXECUTE function
 - 8 - Error in the ODBCCOMMITE function
- *Update command*
 - 9 - Error in the ODBCPREPARE function
 - 10 - Error in the ODBCEXECUTE function
 - 11 - Error in the ODBCCOMMITE function
- *Delete command*
 - 12 - Error in the ODBCPREPARE function
 - 13 - Error in the ODBCEXECUTE function
 - 14 - Error in the ODBCCOMMITE function

Background Tasks

10.7 Math Worksheet

Math worksheets are used to implement program logic (using the internal Scripting Language) that is to be continuously executed during runtime, rather than on specific actions like the user pressing a button on a screen.

Math worksheets are functionally similar to [script on page 324](#) worksheets, except that the internal scripting language is used instead of VBScript.

To create new math worksheets, open the context menu of the "Tasks" folder and select **Add... ► Math worksheet**

To edit an existing math worksheet, double-click on it in the Project Explorer.

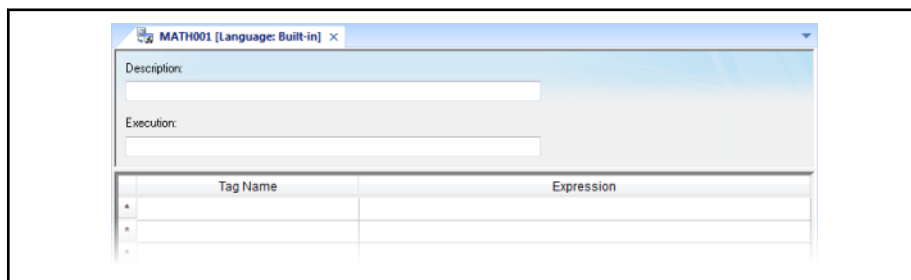


Fig. 10-6: Math worksheet

Math worksheets are divided into two areas:

- "Header" area (top section) containing information for the whole group; and
- "Body" area (bottom section) to define each tag, expression, and programming lines (logical routines and mathematical calculations through functions and logical operations) in the group.

To use the "header" parameters on this worksheet, proceed as follows:

- **Description:** Enter a description of the worksheet for documentation purposes.
- **Execution:** Enter an expression, a single tag or a constant value to determine when the worksheet is to be executed.



The project executes the worksheet only if the **Execution** field result is "not zero". If the worksheet is to be executed always, enter 1 (constant value) into the **Execution** field.

To use the "body" parameters on this worksheet, proceed as follows:

- **Tag Name** field: Enter a tag to receive a return value from the specified calculation in the **Expression** column.
- **Expression** field: Enter an expression to receive a return value from the specified tag in the **Tag Name** column.

To execute a math worksheet at a specific time independently of the configuration in the **Execution** field, use the [Math on page 557](#) function where expressions are allowed.

10.8 Script Worksheet

10.8.1 Script Worksheet, General Information

Script worksheets are used to implement program logic (using VBScript) that is to be continuously executed during runtime, rather than on specific actions like the user pressing a button on a screen.



Script worksheets are functionally similar to [math on page 324](#) worksheets, except that VBScript is used instead of the internal scripting language.

To create new script worksheets, open the context menu of the "Tasks" folder and select **New Script**

To edit an existing script worksheet, double-click on it in the Project Explorer.

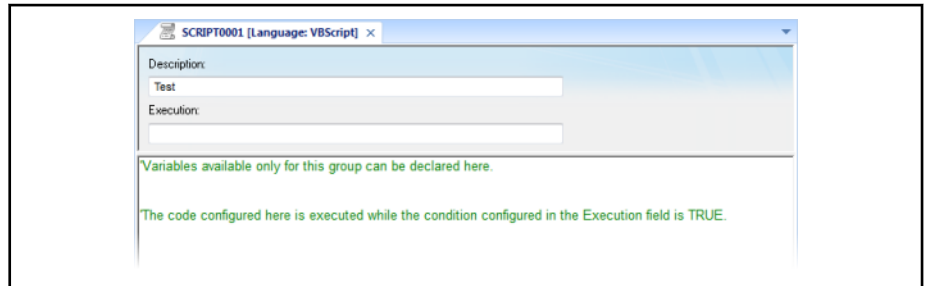


Fig. 10-7: Script worksheet

The code configured in each Script worksheet is executed by the Background task. The project scans the worksheets sequentially (based on the worksheet number) and executes only the groups in which the condition configured in the **Execution** field of the worksheet is TRUE (i.e., non-zero).



The syntax supported by the internal scripting language in the **Execution** field has to be used. VBScript is only supported by the body of the worksheet.

Variables declared in the worksheet have local scope for that specific group only. They are not available for any other VBScript interface.

Procedures (i.e., functions and subroutines) cannot be defined in the script worksheet. However, procedures defined in the [global procedures on page 716](#) or in the [startup script on page 325](#) can be called.

Example:

```
'Variables available only for this group can be declared here  
Dim myVar, myTest  
myTest = 1
```

```
'The code configured here is executed while the condition configured in the Execution field is TRUE  
myVar = $FindFile("c:\*.txt")  
If MyVar > 0 Then  
    $TagNumOfFiles = myVar  
End If
```

NOTICE

If a script worksheet is saved during runtime (online configuration) the startup script is executed again.

The current value of the local variables of all script worksheets are reset.

10.8.2 Script Worksheet Startup

The startup script worksheet is a VBScript interface that is automatically executed when the project is run.

To edit the startup script worksheet, double-click on it in the Project Explorer. (It can be found in the folder "WinStudio\Tasks\Scripts")

Background Tasks

The worksheet is displayed:

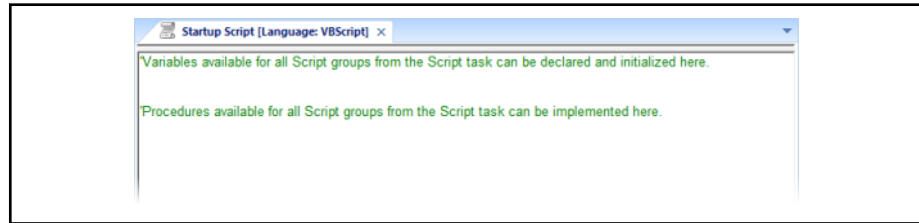


Fig. 10-8: Startup script worksheet

The code configured in this worksheet is executed just once if the Background Task module (BG_Task) is started. This interface is useful for initializing variables or executing logics that have to be implemented when running the project.

Variables can be declared and initialized and procedures defined. However, variables or procedures declared in this interface are available ONLY to the script worksheets executed by the Background Task module. They are not available to any VBScript interface from the Graphic module.

Example:

```
'Variables available for all Script groups from the Script task can be declared and initialized here
Dim MyVar, Counter
MyVar = 100

'Procedures available for all Script groups from the Script task can be implemented here

Function AreaEquTriangle(base, high)
    AreaEquTriangle = (base * high) / 2
End Function

Sub CheckLimits(myValue, myHiLimit, myLoLimit)
    If (myValue > myHiLimit Or myValue < myLoLimit) Then
        MsgBox("Value out of range")
    End If
End Sub

'The code configured here is executed just once when the Background task is started
If $GetOS() = 3 Then
    MsgBox ("Welcome! This project is running under Microsoft Windows Embedded operating system.")
Else
    MsgBox("Welcome! This project Is running under Microsoft Windows desktop operating system.")
End If
```

10.9 Scheduler Worksheet

Scheduler worksheets are used to execute program logic (using the internal scripting language) at a specific date/time, on a regular time interval or upon triggering an event.

To create new scheduler worksheets, open the context menu of the "Tasks" folder and select **Add... ► Scheduler**

To edit an existing scheduler worksheet, double-click on it in the Project Explorer.

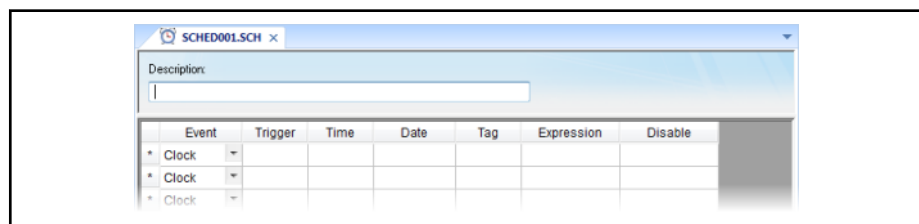


Fig. 10-9: Scheduler worksheet

"Scheduler" worksheets are divided into two areas:

- **Header** area (top section) containing information for the whole group

Background Tasks

- **Body** area (bottom section) to define each tag, expression and condition for the group.

Use the parameters on this worksheet as follows:

- **Description:** Enter a description of the worksheet for documentation purposes.

- **Event**

Selection list: Click to select an event type from the following options:

- **Calendar:** Generates time periods exceeding 24 hours. For example, it is possible to define events that print a report every Friday at a specific time.



Be sure to complete the **Date** field to set a specific date for event execution.

- **Time:** Generates time periods smaller than 24 hours (intervals in minutes or seconds). This function is frequently used with trend graphics. For example, it is possible to define a tag that is incremented each hour.
- **Change:** Event related to the change of a tag in the **Trigger** field.



This only works for tag changes on the project server, regardless of the scope defined for the tag.

- **Trigger:** This field is used only with the **Change** Event type. Enter the name of a project tag into this field. The **Expression** is evaluated, if the value of the tag is changed.

- **Time:** This field is used with the **Calendar** and **Clock** Event types.

If the Event type is **Calendar**, **Time** is a specific time of the day on **Date**. If **Date** and **Time** are reached, **Expression** is evaluated. If **Clock** is selected as Event type, **Time** represents a time interval from project execution. Every time the interval occurs, **Expression** is evaluated.

Either way, enter a time using the "HH:MM:SS.ms" format. Valid values are "00" to "23" for hours, "00" to "59" for minutes, "00" to "59" for seconds, and "1" to "9" for milliseconds. (Milliseconds are optional.) Examples: "03:00:00" is every three hours, "00:00:00.1" is every 100 milliseconds.

- **Date:** This field is used only with the **Calendar** Event type. Enter a specific date in "MM/TT/YYYY" format. If **Date** and **Time** are reached, **Expression** is evaluated.

Valid values are "01" to "12" for months, "01" to "31" for days, and "1900" to "2099" for years. If the field is left blank, the event occurs daily at the specified **Time**.

- **Tag:** Enter a tag that is to receive the value returned by **Expression** (if any).
- **Expression:** Enter an expression to be evaluated. This field is used by all events.
- **Disable:** Contains a disable condition for the specified function. Leave this field blank or use an expression value equal to zero to execute the function. If an expression value equal to one is used, the function is not executed (Disable = 1).

Background Tasks

10.10 Database/ERP Worksheet

In addition to ODBC, "WinStudio" also supports Microsoft .NET ActiveX Data Objects (ADO.NET) for interfacing between the project tags database and other external databases. A database/ERP worksheet is used to associate project tags with external database fields.



For further information about ADO.NET support in "WinStudio" - including how to communicate with remote databases using the "WinStudio" Database Gateway software - please refer to [Database interface on page 429](#).

To communicate with an external database, a [connection on page 328](#) to the database has to be configured first and then, a [worksheet on page 329](#) has to be created that associates project tags with the database fields.

Database connections

To create a new connection to a target database, proceed as follows:

1. In the [Project Explorer on page 55](#), open the context menu of the **External Databases** folder and select the entry **Database Connection Configuration**.
2. The "Database Connection" configuration dialog is displayed listing all already configured database connections.

Press **Add** to create a new database connection. The "Database Connection" dialog is displayed.

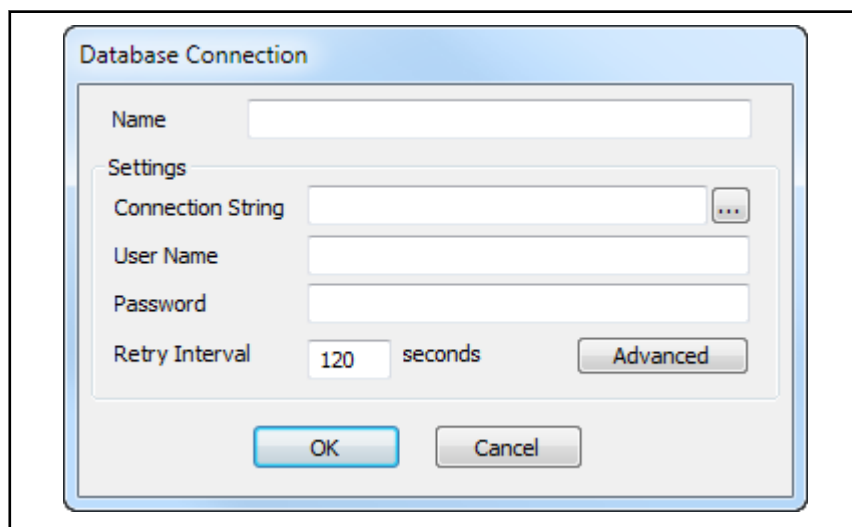


Fig. 10-10: Database Connection dialog

3. In the **Name** field, enter the name to be used to reference the target database. Multiple database connections can be created, however, each connection requires a unique name.
4. In the **Connection String** field, click the browse button ... to open a standard "Data Link Properties" dialog. Use the dialog to configure a connection string for the target database.



The list of database providers displayed in the "Data Link Properties" dialog depends on the providers actually installed and available on the station where the development application is running. For further information about using the "Data Link Properties" dialog, please refer to the Windows Help.

Background Tasks

5. In the **User Name** and **Password** fields, enter an appropriate login for the target database. The login has to have already been created on the database server and requires privileges to read from and write to the database tables.
6. If the connection is established to a remote database via the [Studio Database Gateway on page 432](#), click the **Advanced** button to open the advanced settings dialog.

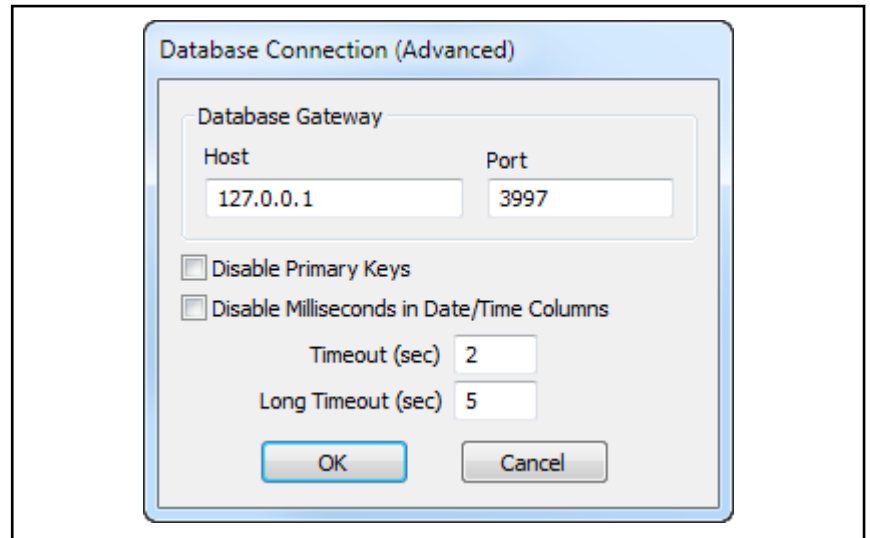


Fig. 10-11: Database Connection (Advanced) dialog

7. In the **Host** field, enter the IP address of the station that is running the "WinStudio" Database Gateway software (**STADOSvr.exe**). In the **Port** field, enter the port number on which the software has been configured to run.

Additional settings to be configured, if necessary:

- **Disable Primary Keys** checkbox: "WinStudio" tries to define a primary key to the table in order to speed up the queries. If a database is used that does not support primary keys (e.g., Microsoft Excel), this checkbox is to be enabled.
- **Disable Milliseconds in Date/Time Columns** checkbox: "WinStudio" tries to include milliseconds when saving a date/time in the database. If a database is used that does not support milliseconds, this checkbox is to be enabled.

8. Click **OK** to save the connection configuration and close the dialog.

Database connections are saved as XML files in the `\project_name\Config` sub-folder. Each file is given the same name as the name of the connection (as entered in the **Name** field of the "Database Connection" dialog) with the **.XDC** file extension. For example, the connection configuration "DB1" is saved in the file...

`\project_name\Config\DB1.XDC`

Database worksheet



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Database worksheets allow asynchronous execution of database operations and offer a user-friendly interface for building SQL commands. To create new

Background Tasks

database worksheets select the entry **New Database Worksheet** in the "Tasks" or "External Databases" context menu.

A new worksheet is displayed, as shown in the following figure:

Fig.10-12: Database worksheet

Database worksheets are saved in the `\project_name\Config` directory, with the `.XDB` file extension. Each new worksheet is automatically numbered in the order of its creation. For example, the first worksheet created is saved in the file...

`\project_name\Config\DB001.XDB`

Database worksheets are executed under the "Database/ERP Runtime" task. However, creating a new worksheet does not automatically activate the task. The task has to be configured in the Task configuration (entry **Tools ► Status** in WinStudio folder context menu) to start during runtime. For further information, refer to <Task execution>.

Database worksheets run only on the server and all triggers have to be configured with server tags.

Worksheet header

The header of the database worksheet is configured as follows:

- **Description:** Enter a description of the worksheet for documentation purposes.
- **Status:** Enter the name of a numeric tag that is to receive status codes for database operations during runtime:

Status codes for external database operations

Status code	Description
4	Result set is empty
3	Cursor released and query successfully closed
2	Beginning of result set reached, usually while moving cursor to the previous row
1	End of result set reached, usually while moving cursor to the next row

Background Tasks

Status code	Description
0	No errors; status normal
-1	Error while connecting to the specified database (see Connection below)
-2	Error while selecting the result set
-3	Error while moving the cursor to the next row (see Next trigger below)
-4	Error while moving the cursor to the previous row (see Previous trigger below)
-5	Error while closing the query (see Close Query trigger below)
-6	Error while inserting rows in the result set (see Insert trigger below)
-7	Error while updating the result set (see Update trigger below)
-8	Error while deleting the result set (see Delete trigger below)

- **Completed:** Enter the name of a numeric tag that is to be toggled when database commands are successfully executed.
- **Error message:** Enter the name of a String tag that is to receive detailed error messages if errors occur during runtime.
- **Connection:** Click to select a connection to the target database. All available connections are listed, as configured in the "Database Connection" dialog [described above on page 328](#).
- **Type:** Click to specify how the result set is to be selected for the worksheet:
 - **Table:** Enter a table name and an optional filter condition. (The filter condition is equivalent to the SQL "Where" clause.) All rows of the table that match the filter conditions are selected.
 - **SQL:** Enter a custom SQL "Select" statement.



For **Table**, **Condition** and **SQL Statement**, the names of project tags that contain the desired information can be entered. This enables programmatically changing the selection during runtime. However, tag names have to be enclosed in curly brackets ("{}") to distinguish them from actual Strings. Additionally, existing selections have to be released before opening new ones; see **Close Query** below.

- *Cursor Triggers pane...*
 - **Selection:** Enter any tag; if the value of the tag changes, a new cursor opens the first row of the result set and copies the values to the tags configured in the worksheet body.
 - **Next:** Enter any tag; if the value of the tag changes, the cursor moves to the next row of the result set and copies the values to the tags configured in the worksheet body.
 - **Advanced:** Click to open the "Advanced Cursor Options" dialog:

Background Tasks

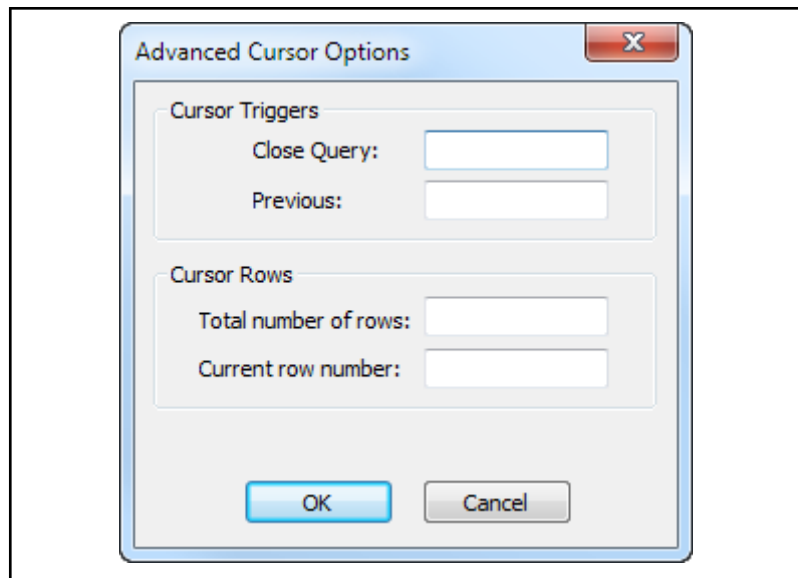


Fig. 10-13: Advanced Cursor Options dialog

- **Close Query:** Enter any tag; if the value of the tag changes, the cursor releases the result set.
 - **Previous:** Enter any tag; if the value of the tag changes, the cursor moves to the previous row of the result set and copies the values to the tag configured in the worksheet body.
 - **Total number of rows:** Enter a numeric tag that is to receive the total number of rows in the result set.
 - **Current row number:** Enter a numeric tag that is to receive the number of the current row (i.e., the position of the cursor). When a result set is first opened using the **Select** trigger, this number is 1. Each **Next** trigger increases this number and each **Previous** trigger decreases it.
- *Table Triggers pane*
 - **Insert** field: Enter any tag; if the value of the tag changes, a new row is inserted with the current values of the tags configured in the worksheet body.
 - **Update** field: Enter any tag; if the value of the tag changes, all rows of the result set are overwritten with the current values of the tags configured in the worksheet body.
 - **Delete** field: Enter any tag; if the value of the tag changes, all rows of the result set are deleted.



Table triggers are available only if **Type** is set to **Table**, as these operations work on the entire table row.

Worksheet body

In the body of the worksheet, [application variables on page 106](#) can be mapped to the columns (fields) of the result set. For each row of the body, enter a **Tag Name** and its corresponding **Column**. The available columns depend on how the result set is selected.

This selection may change during runtime. For this reason, it is crucial to map all available columns.

11 Communication with Other Devices

11.1 Communication with Other Devices, General Information

Communication tasks/worksheets are used to exchange tag values with remote devices (e.g., PLCs) and other "WinStudio" projects and any system that implements supported protocols like OPC and DDE.

Communication of tags with a data source is achieved by means of communication drivers and the respective worksheets in the **WinStudio** ► **Variables** ► **Communication** ► **<Driver name>** folder.

11.2 Configuring Direct Communication with a Remote Device

11.2.1 Configuring Direct Communication with a Remote Device, General Information

Communication drivers are DLLs implementing the communication protocol of a specific remote device to read and write its data. WinStudio offers various standard communication drivers. For communication with VPx and VEx devices, the BR_WS driver is additionally available. For numerous devices, additional drivers are available that can be added to systems to exchange data with these devices.

In the properties dialog of drivers, the communication interface between the project and remote devices can be defined.



Consult the **Help** menu for a description of the default functions and characteristics for all drivers. If a project is to be developed, also refer to the specific documentation provided with each communication driver. This documentation can usually be found in the **DRV** directory.

For communication driver configuration, the interface parameters (for example, the station address and the baud rate) as well as device addresses are to be specified and then linked to project tags.

To add or remove drivers, proceed as follows:

- Open the context menu of the folder **WinStudio** ► **Variables** ► **Communication** and select the entry "Add/Remove Driver"

The following dialog with a list of available drivers is displayed.

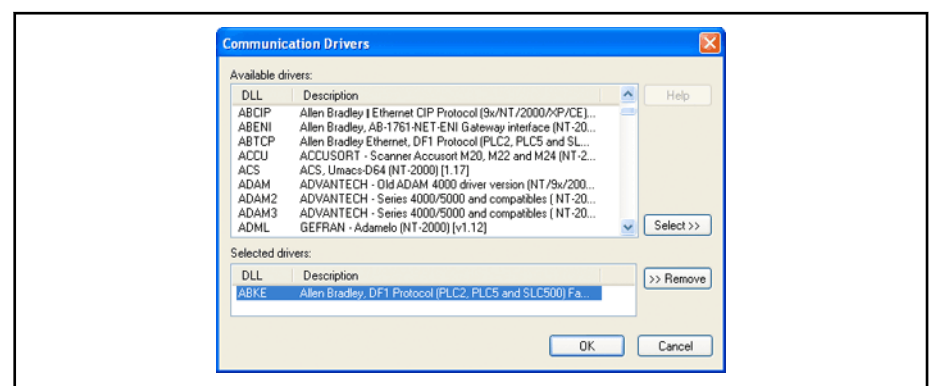


Fig. 11-1: Communication Drivers dialog

Communication with Other Devices

The properties in this dialog have the following function:

- **Available drivers:** Lists all available drivers and a brief description of each
- **Help:** With the **Help** button, a detailed description for driver configuration can be opened for the drivers selected under "Available drivers" or "Selected drivers". If additional drivers are installed, no help may be available for these. In this case, the "Help" button is deactivated.
- **Select:** Click on this button to use the driver selected in the **Available drivers** field in the project.
- **Selected drivers:** Lists all selected drivers and their descriptions (if available).
- **Remove:** Click on this button to stop using the driver selected in the **Available drivers** field in the project and to remove it from the project.

Changed settings can be applied by pressing **OK**. In the "Communication" folder, a respective subfolder for added drivers is created or removed.

The context menu of a driver subfolder contains the entry "Properties". This entry opens the configuration dialog for the respective driver to configure the following settings:

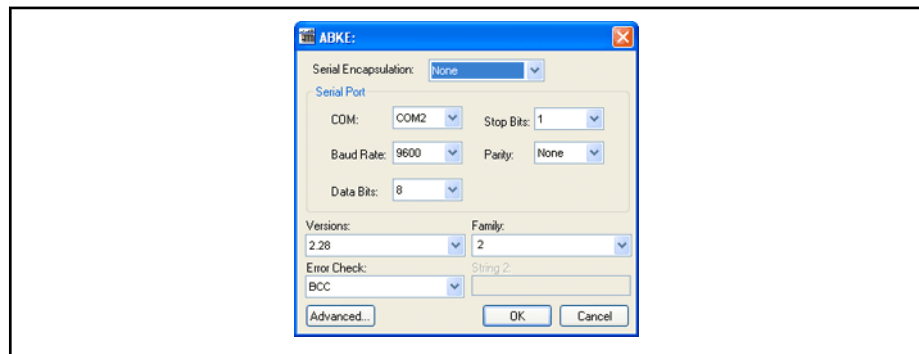


Fig. 11-2: Example of Communication Parameters dialog

The individual driver settings have the following meaning:

- **Serial Encapsulation:** Enables serial drivers to communicate with modem, TCP/IP or UDP connections. This setting is only supported for serial drivers developed with the UNICOMM library. This applies to most serial drivers available in this product.



This section covers only the **"None"** option, which enables the driver to connect via a normal serial channel. Please refer to "Using TCP/IP and UDP encapsulation" and "Using modem connections" below for more information about other encapsulation modes. "Serial encapsulation tests" below lists drivers that have been tested with modem, TCP/IP and UDP modes.

- **COM:** Select a serial communication port.
- **Baud Rate, Data Bits, Stop Bits and Parity:** These parameters are required for the configuration of a serial port.
- **Long1, Long2, String1, and String2:** These settings are driver-specific. In the example above, the driver uses **Long1** to set up the error detection method and **String1** to define the PLC family type.
- **Advanced:** Click on this button to open a dialog to configure advanced settings: Use this dialog to change the default driver parameters.

Communication with Other Devices

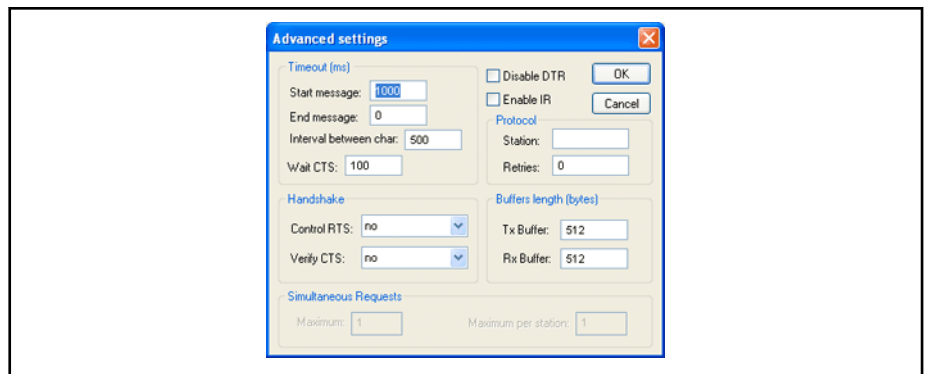


Fig. 11-3: Advanced settings dialog

To specify or change default driver parameters, proceed as follows:

- *Timeout (ms) pane:*
 - **Start message:** Specify the timeout for the message start.
 - **End message:** Specify the timeout for the message end.
 - **Interval between char:** Specify the timeout between each character.
 - **Wait CTS:** Specify the timeout for the "Clear to Send" (CTS) waiting time.
- *Handshake pane:*
 - **Control RTS:** Specify whether to use the "Request to Send" (RTS) control.
 - **Verify CTS:** Specify whether to use the "Clear to Send" type of verification.
 - **Disable DTR:** Click (enable) this checkbox to disable the DTR (data terminal ready) function (the DTR signal is not set by the driver before starting the communication).
 - **Enable IR** ("only available on Windows Embedded target systems"): Click (enable) this checkbox to enable the serial driver to use an infrared interface (COM2 port) instead of a standard serial port to communicate with the device (such as the PLC, I/O, handheld computers, etc.).
- *Protocol pane:*
 - **Station:** Some slave drivers, such as the Modbus Slave (MODSL), require a slave network address. Use this field to specify the slave address.
 - **Retries:** Enter a numeric value to specify how many times the driver attempts to execute the same communication command before considering a communication error for this command.
- *Buffers length (bytes) pane:*
 - **Tx Buffer:** Specify the transmission buffer length (in bytes):
 - **Rx Buffer:** Specify the reception buffer length (in bytes).
- *Simultaneous Requests pane (available only for selected drivers):*
 - **Maximum:** Specify the maximum number of requests that may be sent simultaneously to all connected devices.
 - **Maximum per station:** Specify the maximum number of requests that may be sent simultaneously to a single device.

Communication with Other Devices



The maximum number of simultaneous requests depends on the device and protocol specifications. Please consult the device manufacturer's documentation.

The development application provides two interfaces to configure the driver (associating project tags to device addresses):

- **<MAIN DRIVER SHEET>** Provides the easiest method for communication configuration between project tags and device addresses. With this interface, tags can be automatically grouped to provide the best performance during runtime. This interface cannot be used to control the time needed to scan a group of tags individually.
- **STANDARD DRIVER SHEETS** on page 339 Enables controlling the time needed to scan a group of tags individually.

Both sheets can be used at the same time.

Using TCP/IP and UDP encapsulation

Most of the serial drivers allow the use of TCP/IP or UDP/IP encapsulation. The encapsulation mode has been designed to provide communication with serial devices connected to terminal servers on Ethernet or wireless networks. A terminal server can be seen as a virtual serial port. It converts TCP/IP or UDP/IP messages on ethernet or wireless networks to serial data. Once the message has been converted to a serial format, standard devices that support serial communication to the terminal server can be connected.

The following figure illustrates one example of applying this solution:

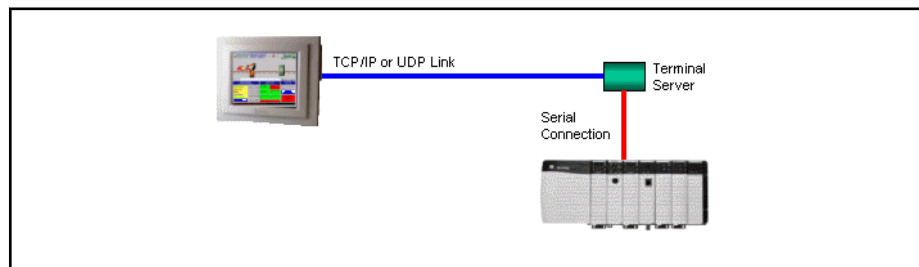


Fig.11-4: TCP/IP encapsulation

To enable encapsulation, proceed as follows:

1. Right-click on the Driver folder and select **Settings** from the context menu. By this means, the communication parameters are accessed.
2. In the **Serial Encapsulation** field, select **TCP/IP** or **UDP/IP**:

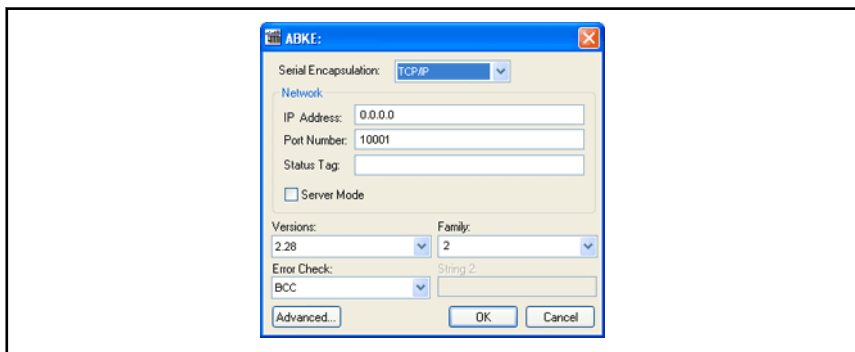


Fig.11-5: Dialog: Encapsulation

The following fields are available:

- **IP Address:** Specify the IP Address for the terminal server. In this field, tags can be entered between curly brackets.

Communication with Other Devices

- **Port Number:** Enter the TCP/IP or UDP/IP port number.
- **Status Tag:** This field is available only when using TCP/IP. The tag on this field receives the value 1 when the TCP/IP connection is established; otherwise, it receives 0.
- **Server Mode:** TCP/IP encapsulation enables the Server mode, making the remote client responsible for establishing the connection to enable the communication.

Using modem connections

Most of the serial drivers enable the use of modem connections. Modem connections have been designed to enable communication with remote serial devices connected via phone lines.

The following figure illustrates one example of applying this solution:

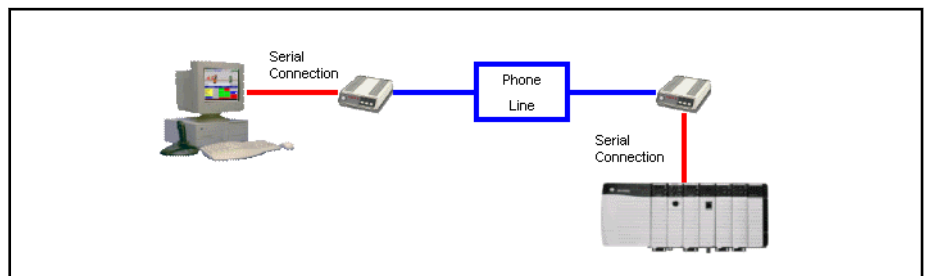


Fig.11-6: Modem connection

To enable the modem connection, proceed as follows:

1. Right-click on the Driver folder and select **Settings** from the context menu. By this means, the communication parameters are accessed.
2. In the **Serial encapsulation** menu, select **Modem**:

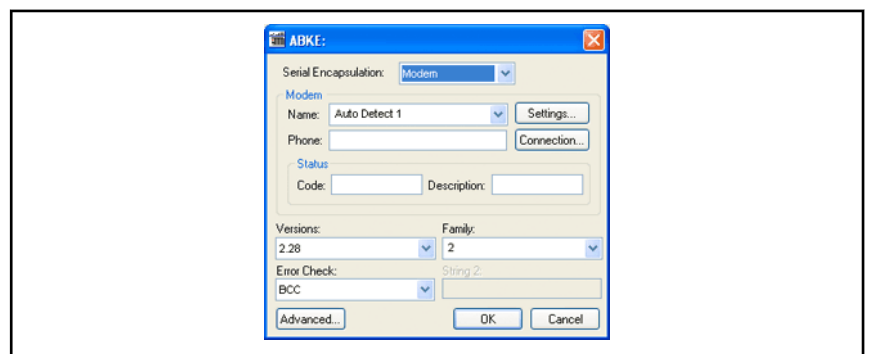


Fig.11-7: Dialog: Serial encapsulation

NOTICE

The Modem option is not supported for Pocket PC v3.00 or older.

The following settings are available:

- **Name:** Select the modem that the driver is to use to establish the connection.
If the modem name is unknown, use the Auto Detect option:
 - **Auto Detect 1** uses the first modem available
 - **Auto Detect 2** uses the second modem available
 - **Auto Detect 3** uses the third modem available
 - **Auto Detect 4** uses the fourth modem available

Communication with Other Devices

- **Phone:** Enter a phone number for the driver to connect to the remote device. In this field, tags can be entered in curly brackets to change the phone number during runtime.
- **Settings:** Click on this button to configure the modem settings. The window displayed after clicking on this button depends on the operating system and the modem type.

NOTICE

The settings configured by clicking on this button are not saved with the project.

The information is saved on the operating system registry and is valid only on the computer that is interacted with. If the project is installed on a different computer, these settings have to be reconfigured.

- **Connection:** Click to open the "Connection Control"

The default connection settings are sufficient for most projects. However, the connection can be fully controlled and incoming calls can be enabled by clicking on this button.

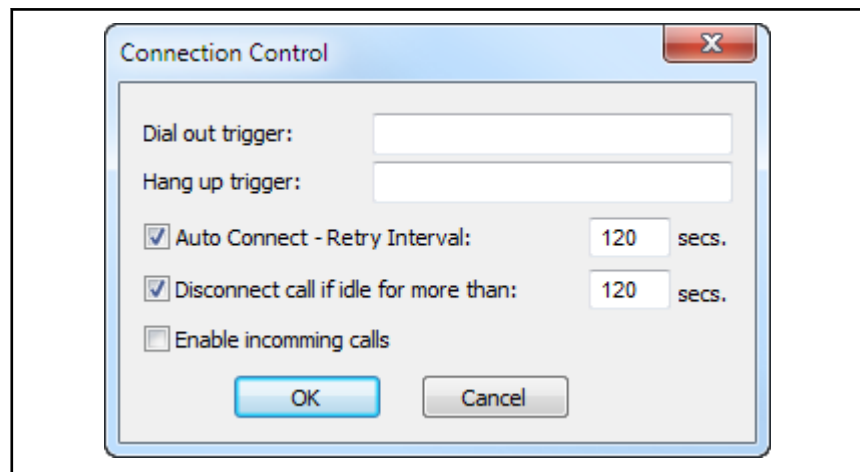


Fig. 11-8: Connection Control dialog

- **Dial out trigger:** If the value of the tag configured in this field changes, the driver tries to connect to the remote device. If the connection has already been established, the command is ignored. This field can be ignored if "Auto Connect" is used.
- **Hang up trigger:** If the value of the tag configured in this field changes, the driver disconnects from the remote device. If the device is disconnected, the command is ignored. This field can be ignored if "Disconnect call if idle for more than" is enabled.
- **Auto Connect:** If this option is enabled, the driver tries to connect to the remote device before sending any information. If the connection fails, the next attempt is made after the Retry Interval has expired.
- **Disconnect call if idle for more than:** If this option is enabled, the driver automatically disconnects from the remote device if no communication takes place after the specified time.
- **Enable incoming calls:** Enable this option to enable the driver to receive calls from the remote device. Hang up trigger can be used to drop calls once they have been established. Please note that drivers can use both incoming and outgoing calls.

- *Status pane*
 - **Code:** Enter a tag to receive the status value of the connection.
The status values have the following function:
 - 0 = Disconnected
 - 1 = Connected
 - 2 = Dialing
 - 3 = Dropping
 - 4 = Closing Line
 - **Description:** Enter a tag that is to receive a complete description of the current status. The description is associated with the **Code** field; however, it provides additional information about the current status.

Serial encapsulation tests

Most serial drivers work with every serial encapsulation mode. However, most of the drivers had been developed before the encapsulation modes were created. The following table lists the drivers tested with specific encapsulation modes. If a driver is to be used that is not listed and if it is not sure if it works, please contact the distributor.

Driver	Modem	TCP/IP	UDP/IP
MODSL	X	X	X
ABKE	X	X	X
MODBU	X		X
OMETH	X		

X = Item has been tested

11.2.2 Driver Sheets

Multiple driver sheets can be created for each driver to enter tags and to link them to an address on the remote device. Additionally, each driver sheet offers various configuration options to control execution during runtime.

To create a new driver sheet, open the context menu of the Driver folder and select "New Driver Sheet". Each driver sheet has a unique number that can be changed at any time by means of the "Rename" context function. To open driver sheets, double-click on the new element in the Driver folder.

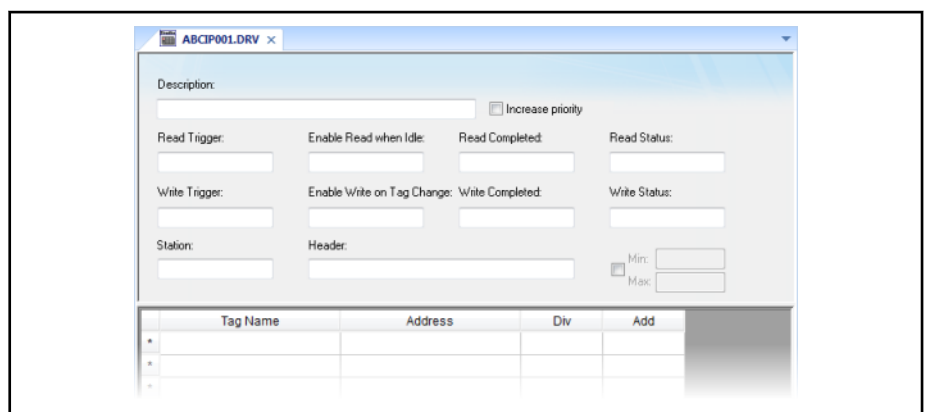


Fig. 11-9: Driver sheet editor

Communication with Other Devices

The driver sheet editor is divided into two sections:

- The "header" area (top section) contains parameters that apply to the communication of all tags configured in the body area of the respective worksheet.
- The "body" area (bottom section) links tags available in the project with device addresses.

To use the "header" parameters, proceed as follows:

- **Description:** Enter a description for documentation purposes.
- **Increase priority:** Enable this option if this driver sheet is to be processed first in every execution cycle.

NOTICE

If the Increase priority option is enabled, special attention has to be given to this worksheet.

If the worksheet keeps triggering communication commands, the project may never be able to execute the other driver sheets.

- **Read Trigger:** Enter a tag if a read command for all tags is to be executed in the body area for every value change of the tag.
- **Enable Read when Idle:** Enter a tag or constant. Use a tag value (or constant) greater than zero to enable reading from the equipment.

NOTICE

If a constant (not zero) is used, check whether the project requires continuous reading.

Continuous reading may affect execution of the project.

- **Read Completed:** Enter a tag to monitor execution of the read command. The communication driver changes the tag after each "read" command.
- **Read Status:** Enter a tag that is to be updated with the [status on page 757](#) of the last "read" command by the communication driver.
- **Write Trigger:** Enter a tag if a write command for all tags is to be executed in the body area for every value change of the tag.
- **Enable Write onTag Change:** Enter a tag value or constant (not zero) to activate the communication driver to continuously check the tag value for changes. If a change occurs at this tag value, this value is written by the project to an address in the remote device.
- **Write Completed:** Enter a tag that is to be updated by the communication driver if a "write" command is completed.
- **Write Status:** Enter a tag that is to be updated with the [status on page 757](#) of the last "write" command by the communication driver.
- **Station:** Enter the device station number within the network. The syntax in this field varies according to the communication driver. For further information, refer to the documentation of the respective driver.



If the driver is configured for serial encapsulation via TCP/IP or UDP/IP, the station can be specified for some drivers by means of the following format:

IP_address:port_number|station

For example:

10.169.25.18:1234|Station5

For information whether this function is available for the selected driver, please refer to the documentation of the respective driver.

- **Header** field: Specify the data type and/or initial address to read or write in the device. The syntax in this field varies according to the communication driver. For further information, refer to the respective <driver documentation>.



In the **Station** and **Header** fields, text with tag values using the **text {tag}** syntax can be entered.

- **Min** and **Max**: Enable this option to specify the minimum and maximum values for remote device data.
- **Min** and **Max** (become active only if the **Min** and **Max** checkbox is enabled): Enter a range of values to be converted into an engineering unit. These fields determine the minimum and maximum range of values. These values affect all tags in the worksheet. For example, the memory contains values from 0 to 4095, i.e., 0% to 100% in the user interface. In this example, 0 to 100 has to be specified for the min and max tag parameters.

Use the "body" area parameters as follows:

- **Tag Name**: Enter the name of a tag to be linked to the device address.
- **Address**: Enter a remote device address (or address offset) related to the project tag. The syntax in this field varies according to the communication driver. For further information, refer to the documentation of the respective driver.
- **Div**: Specify a division constant to use in case scale adjustment is required. The project uses this value as a division factor in a read operation and a multiplication factor in a write operation. Do not use this field if **Min** or **Max** is used in the configuration body.
- **Add**: Specify an addition constant to use in case scale adjustment is required. The project uses this value as an addition factor in a read operation and a subtraction factor in a write operation. Do not use this field if **Min** or **Max** is used in the configuration body.



The maximum number of tags in each driver communication worksheet is 512. For some drivers, this number may be smaller. For further information, refer to the <driver documentation>.

For read operations, the following applies:

tag = (value in the device) / Div + Add

For write operations, the following applies:

value in the device = (tag - Add) * Div

Communication with Other Devices

If the cells in the **Div** and **Add** fields are left empty, this function is ignored.

11.3 Configuring an OPC Client Connection to an OPC Server

OPC worksheets are used to communicate with any system that implements the OPC Server protocol.

The OPC client implements the OPC standard as described in document "OLE for Process Control Data Access Standard Version 1.0A". This document is available on the [OPC Foundation website](#).



Before using the OPC client task/worksheet in the project, make sure that the OPC server software is properly installed and configured on the system to which the connection is to be established.

To configure a new connection, insert a new OPC worksheet in the folder **WinStudio ► Variables ► Communication ► OPC** in the Project Explorer.

Tag Name	Item	Scan	Div	Add
*		Always		
*		Always		
*		Always		

Fig.11-10: Example of an OPC worksheet

To configure an OPC driver sheet, use the following parameters:

- **Description:** Enter a description of the OPC worksheet for documentation purposes.
- **Server Identifier:** Enter the name of the server to connect to. If the server is already installed on the computer, the server name can be selected from the list.
- **Disable:** Enter a tag or a constant with a value other than "0" to disable communication with the OPC server. Enter "0" or leave the field blank to enable communication.
- **Read Update Rate (ms):** Specify how often the server is to update this group (in milliseconds). Enter "0" to specify the fastest practical rate for the server.
- **Percent Deadband** (available for analog items only): Specify how much percent change in an item value is to cause a notification by the server.
- **Status:** Enter the name of a tag to receive the status of the connection. The status "GOOD" corresponds to a value of 1.
- **Remote Server Name:** Node name or IP address of the server on the node network.
- **Read before writing:** If this option is enabled, the original values are read from the OPC server directly before writing new variable values to the server. The project does this by first buffering the new values to be written and then reading the original values from the server. The new

Communication with Other Devices

values are not written from the buffer to the server before the project is synchronized with the server.

- **Read after writing:** If this option is enabled, the values from the OPC server are read back directly after writing new variable values to the server.

NOTICE

The "Read before writing" and "Read after writing" options are offered as the OPC client/server specifications define that the value of an element on the client – in this case, the project – is not to change unless the change is initiated by the server. This way, the client always stays in sync with the server.

The project, however, may be designed to change the values according to runtime processes or user input. For this reason, the best way to change the values while staying in sync with the server is to make it seem like the changes are initiated by the server. If both options are enabled, the following sequence of events is performed on every scan of the OPC worksheet:

1. The new values on the client are buffered.
2. The original values on the server are read by the client, i.e., the client is synchronized with the server.
3. The new values are written from the buffer to the server.
4. The new values on the server are read by the client, i.e., the client is synchronized with the server again.

At the end of each reading/writing cycle, the values reflect the project status even though, technically speaking, the project is just staying in sync with the server.

In most projects, both options are to be enabled. In some projects, however, this may cause items to bounce between the original values and the new values. If this is a problem, try moving those items to another OPC worksheet where the **Read before writing** and **Read after writing** options are disabled.

- **Accept Tag Name in the Item column:** If this option is enabled, the text enclosed in curly brackets in the "Item" field is interpreted as a Tag Name (String type). In this case, the value of this tag is used as item name by the OPC server. This enables changing the link between a tag and an OPC item during runtime by changing the values of the configured tag. If the **Accept Tag Name in the Item column** option is disabled, all characters configured in the "Item" column are considered part of the item name (including the characters in curly brackets).
- **Tag Name:** Enter the names of tags to be linked to the server items.
- **Item:** Enter the name of the server item. After selecting the OPC Server, items can be selected from the Server using the OPC Browser. Right-click in the **Item** field and select the **OPC Browser** option.



In this field, tag names can be entered between curly brackets (e.g., {TagName}). This way, item names can be changed dynamically during runtime.

Communication with Other Devices

- **Scan:** Specifies the condition under which the tag value is read from the remote device or server and then updated in the project database. For this, the following options apply:
 - **Always** means the tag is read and updated during every scan of the communication worksheet, regardless of whether the tag is used in any other WinStudio screens, scripts, or worksheets. This option is recommended for tags that have to be monitored continuously in the background, such as tags that trigger alarms, tags used in recipes, tags that are recorded in the historical database, etc.
 - **Screen** means the tag is read and updated only if it is being used in at least one open screen, either locally or on another client station. This option is recommended for tags that are exclusively used in screens as these do not need to be updated frequently. Selecting this option can improve project performance.
 - **Auto** means the project automatically selects either **Always** or **Screen**, depending on where the tag is used in the project. If the tag is only used in one WinStudio screen, the scan is set to **Screen** by default. If the tag is used in different worksheets (e.g., script, math, alarm, trend, recipe, report, scheduler), the scan is set to **Always** during runtime.

If it is not clear, which option is to be selected, select **Always**. This guarantees that the tag is read and updated.
- **Div:** Specifies the division constant if scale adjustment is required. This value is a division factor in a read operation and a multiplication factor in a write operation.
- **Add:** Specifies the addition constant if scale adjustment is required. This value is an addition factor in a read operation and a subtraction factor in a write operation.

To execute the OPC Client task during project runtime, open the task configuration (**Tools ► Status** in the "WinStudio" folder context menu) and select the default setting "Automatic" under "OPC Client Runtime". In this dialog, the task can also be started manually for test purposes by clicking on **Start**. After running this program, a small icon is displayed in the system tray.

To close the OPC Client runtime task, right-click the icon in the system tray and click **Exit**.



"WinStudio" and CEView also provide an OPC Server communication task named "Studio.Scada.OPC". This task starts automatically if any OPC client (local or remote) attempts to connect to the "Studio.Scada.OPC" server. An OPC client can exchange data with the tags database (project tags, system tags, and shared tags) using the OPC interface.

In addition, the OPC Server task can be started automatically on running the project. Select the OPC server task in the task configuration (**Status** entry in the WinStudio context menu), click on **Start** and select **Automatic**.

OPC Debugging

If the use of OPC causes problems establishing the communication, the messages in the LogWin are to be checked first.

If the project is executed on a Windows Embedded target system, there are two ways to check the log:

1. Remote LogWin

2. Local log

For information about these logs, refer to [Using the LogWin task on page 389](#).

If error messages are found in the log, look them up in the manual/help system and follow the documented steps for solving the problems. (Use <CTRL>+ <F> to search the manual or use the Index to search in the context-related help system.)

To request technical support from the distributor, please provide the following information:

1. Log file
2. Software vendor and product name of the OPC server/client that is used
3. If possible, a copy or an evaluation version of the OPC server for testing purposes
4. The contact information of the technical support of the OPC server/client

Three possible errors and their solutions are listed below...

Security

Error code: **16#80070005 or -2147024891**

Error cause: If the OPC client tries to connect to the OPC server, the DCOM layer usually requires authentication. The computer that is running the OPC server needs to recognize the user logged on to the OPC client computer, and the respective user requires access rights to the OPC server.

Solution: The first step is to create a user on both computers that has Administrator privileges and the same password. Log on with this user to both computers and then try to establish the connection.

Resolving name

Error: **Could not connect to AdviseSink, error: -2147022986 (16#80070776)**

Error cause: There is a problem resolving the computer name.

Solution: This problem can be solved by specifying the IP address of the server instead of specifying the computer name.

Proxy for Windows CE

Error: **OPC server: IIndCP::Advise - Could not query callback interface: 16#80040155**

Error cause: The Windows Embedded device is missing the **OPC-COMN_PS.dll**.

Solution: Download the .dll to the device and register it. The .dll is with the "WinStudio" version, most probably at

[...]\WinStudio\Redist\Wince x.x\processor\

If the .dll cannot be found in the folder of the respective processor, contact the "WinStudio" distributor.

11.4 Configuring an OPC UA Client Connection to an OPC UA Server

OPC UA driver sheets are used to communicate with any system that implements the OPC server protocol.

Communication with Other Devices

About OPC UA The new OPC UA log (OPC Unified Architecture) was introduced by OPC Foundation. For the introduction of the new log, OPC Foundation provided the following information:

" The existing OPC COM based specifications have served the OPC community well over the past 10 years, but as technology moves on so must our interoperability standards. Here are the factors that influenced the decision to create a new architecture: "

- Microsoft has deemphasized COM in favor of cross-platform capable web services and SOA (Service Oriented Architecture)
- OPC vendors are looking for a single set of services to expose the OPC data models (DA, A&E, HDA ...)
- OPC vendors are planning to implement OPC on non-Microsoft systems, including embedded devices
- Other collaborating organizations need a reliable, efficient way to move higher level structured data

OPC UA is intended to be a platform and language-independent protocol that is also backwards-compatible with "classic" OPC systems. For more information, refer to <http://www.opcfoundation.org/UA/>.



This function includes cryptographic software written by Eric Young (eyay@cryptsoft.com).

Configuring a new OPC UA connection

To communicate with a device via the OPC UA server, the connection to this server has to be configured first. To do so, proceed as follows:

1. Open the context menu of the "OPC UA" folder and select the entry "OPC UA Connection Configuration".

A dialog is opened displaying all previously created configurations. To create a new connection to an OPC UA server, press "Add".

The "UA Server Configuration" dialog is displayed:

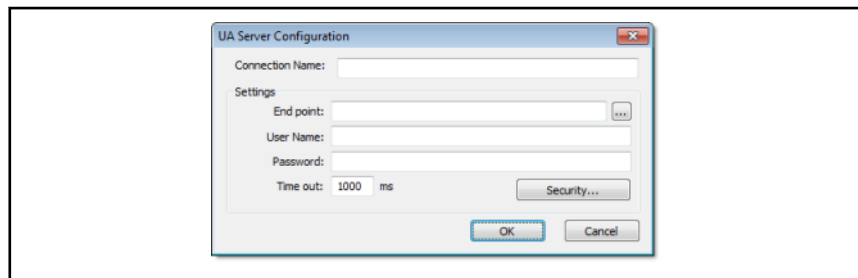


Fig. 11-11: UA Server Configuration dialog

2. In the **Connection Name** box, enter a name for the connection. This name is displayed in the OPC UA Connections folder in the Project Explorer and is the name to be looked for when configuring the OPC UA client worksheet.
3. In the **End point** box, enter the URL of the OPC UA server to be connected to. If the URL is unknown, click the browse button to the right of the box. The "UA Discovery" dialog is displayed:

Communication with Other Devices

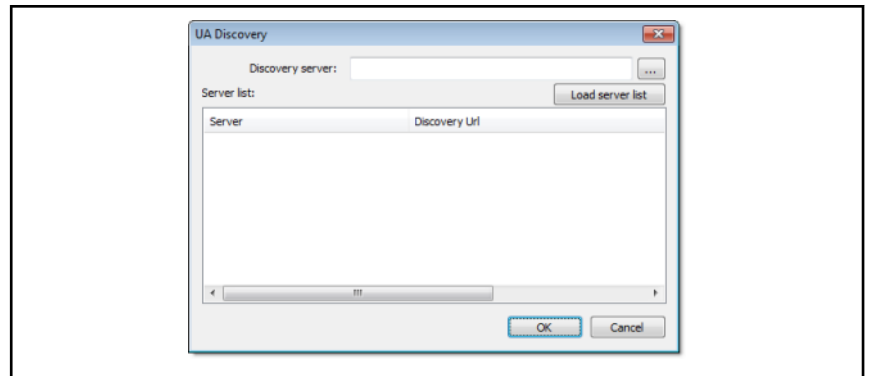


Fig. 11-12: UA Discovery dialog

Use this dialog to find the discovery server, which publishes a list of OPC UA servers on the network and select the server to connect to.

4. In the **User Name** and **Password** field, enter the login data for the OPC UA server.
5. If the OPC UA server is configured to require a secure connection, additional steps are required to install the server certificate in the project and to create a client certificate for the installation on the server:
 - 5.1 Acquire the server certificate and then save it in the "Config" subfolder of the project folder (e.g., `[...]\My Documents\WinStudio Projects\project_name\Config\`).
How the server certificate is acquired depends on the respective server. Please contact the server manufacturer.
 - 5.2 In the "UA Server Configuration" dialog, click **Security**. The "OPC UA Security" dialog is displayed.
 - 5.3 In the **Server Certificate** list, select the server certificate. (All certificates saved in the "Config" subfolder are listed here.)
 - 5.4 In the **Security Policy** and **Message Security Mode** lists, select the appropriate options for how the OPC UA server is configured. Please consult the server manufacturer.
 - 5.5 Click on **Generate Certificate**. The client certificate is generated and saved in the "Config" subfolder.
 - 5.6 Click on **OK** to close the "OPC UA Security" dialog.
 - 5.7 Install the client certificate on the OPC UA server. The precise procedure depends on the respective server. Please contact the server manufacturer.

For further information, download the white paper ["The OPC UA Security Model for Administrators"](#) by OPC Foundation.

6. Click on **OK**. The connection is saved in the OPC UA Connections folder in the Project Explorer.

Configuring a new OPC UA connection

To create a new driver sheet for the OPC UA client, proceed as follows:

1. Open the context menu of the "OPC UA" folder and select "New Driver Sheet".

A new element is created in the OPC UA folder. Now, double-click the driver sheet editor to open it:

Communication with Other Devices

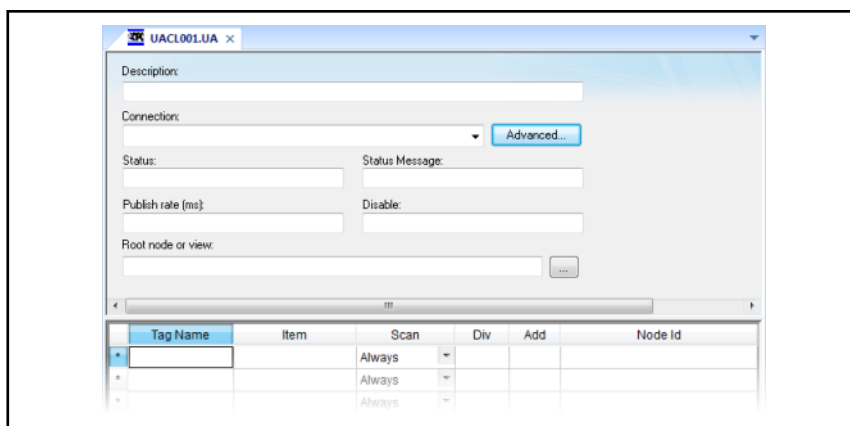


Fig. 11-13: OPC UA client worksheet

2. In the **Description** field, enter a description of the worksheet. This is for documentation purposes only and does not affect the execution of the worksheet.
3. In the **Connection** list, select the server connection configured earlier.
4. For further connection settings, click on **Advanced**. The advanced settings dialog is displayed:

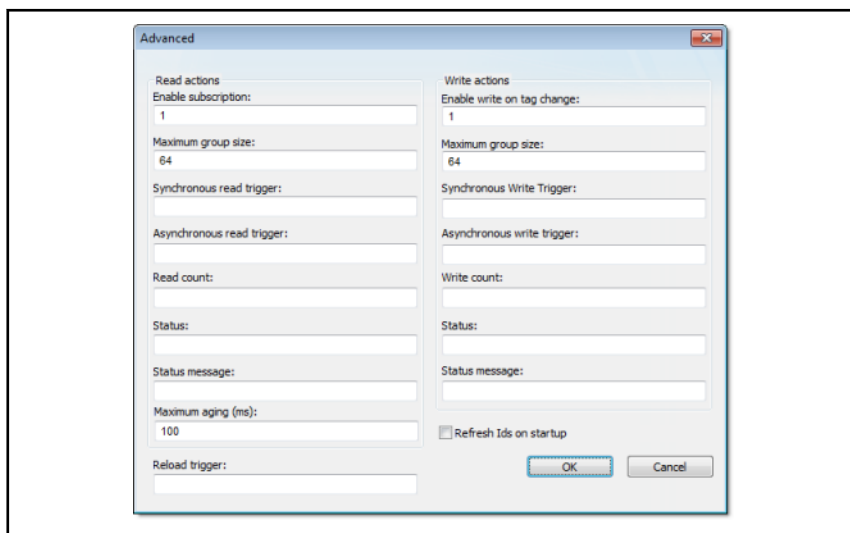


Fig. 11-14: OPC UA client worksheet, Advanced

Review the options and configure as needed:

Communication with Other Devices

Area / Element		Description
Read actions	Enable subscription	If this value is TRUE (not zero), the client constantly requests updates from the server. This option is enabled by default. If it is disabled, read triggers (see below) have to be used.
	Maximumgroup size	The maximum number of tag reads that may be performed in a single read operation. For example, if 1000 items/rows are configured in the worksheet and the Maximum group size is set to 100, 10 read operations are performed during each scan of the worksheet.
	Synchronous read trigger	If the value of this tag/expression changes, the worksheet is scanned and all tag values are read from the server. The project waits for the scan to complete before continuing.
	Asynchronous read trigger	If the value of this tag/expression changes, the worksheet is scanned and all tag values are read from the server. The project continues running without waiting for the scan to complete.
	Read count	The name of a tag (Integer type) that receives a count of the number of read operations performed since the project was run.
	Status	The name of a tag (Integer type) that is to receive a status code for the last read operation performed by a trigger.
	Status message	The name of a tag (String type) that is to receive the corresponding status message.
	Maximum aging	The maximum age (in milliseconds) of values that are to be accepted from the server cache. If a value is older than this, the server is forced to get the latest value from the target device.
Write actions	Enable write on tag change	If this value is TRUE (not zero), a write operation is automatically performed whenever the value of the project tag changes. Note: This option is enabled by default. If it is disabled, write triggers (see below) have to be used.
	Maximumgroup size	The maximum number of tag writes to be performed in a single write operation. For example, if 1000 items/rows are configured in the worksheet and the Maximum group size is set to 100, 10 write operations are performed during each scan of the worksheet.
	Synchronous Write Trigger	If the value of this tag/expression changes, the worksheet is scanned and all tag values are written to the server. The project waits for the scan to complete before continuing.
	Asynchronous write trigger	If the value of this tag/expression changes, the worksheet is scanned and all tag values are written to the server. The project continues running without waiting for the scan to complete.
	Write count	The name of a tag (Integer type) that is to receive a count of the number of write operations performed since the project was run.
	Status	The name of a tag (Integer type) that is to receive a status code for the last write operation performed by a trigger.
	Status message	The name of a tag (String type) that is to receive the corresponding status message.

Communication with Other Devices

Area / Element	Description
Reload trigger	Indirect tags (e.g., {MyTag}) configured in the body of the worksheet are only reloaded if the value of this tag/expression changes.
Refresh IDs on startup	If this option is enabled, the node IDs in the worksheet are refreshed from the specified item path every time the project is run. Note: The startup process of the project may be longer due the ID refreshing process. When activating this option, also specify a root node (see below) to limit how much of the server element list is to be scanned.



Normally, the "WinStudio" OPC client does not apply the Trigger mode that is defined in the OPC protocol. Instead, it allows any change in any tag/expression to be used as a trigger. If the Trigger mode is to be used, configure one worksheet to read the trigger values and configure another worksheet that specifies the read values as triggers values.

5. Click on **OK** to close the "Advanced" dialog and return to the worksheet.
6. In the **Status** field, enter the name of a tag (Integer type) that is to receive connection status codes during project runtime.
7. In the **Status Message** field, enter the name of a tag (String type) that is to receive the corresponding status messages.
8. In the **Publish rate** field, enter the frequency (in milliseconds) at which the client is to request updates from the server.
9. In the **Disable** field, enter a tag/expression. If the value is TRUE (not zero), the worksheet is not executed.
10. In the **Root node or view** field, specify the server node that is to serve as the root for all items in the worksheet body. Specifying a root node makes it easier to find items and improves runtime performance.
11. For each row of the worksheet body, the following information is to be specified:
 - **Tag Name:** The name of a tag. If indirect tags (e.g., **{MyTag}**) are to be used, the **Reload trigger** option described above has to be enabled.
 - **Item:** The server item the tag is to be associated with. To browse the server list of items, right-click in this field and select **Browse** in the context menu.
 - **Scan:** If **Always** is selected, the row is continuously scanned during runtime. If **Screen** is selected, the row is scanned only in case a WinStudio screen that uses the specified tag is opened.
 - **Div:** For scaling – the value is divided by this number when reading from the server and multiplied by this number when writing to the server.
 - **Add:** For scaling – this number is added to the value when reading from the server and subtracted from the value when writing to the server.
 - **Node ID:** The node ID is automatically generated from the full path of the server item.
12. Save and close the worksheet.

Enabling the OPC UA client task

To enable the OPC UA client task for runtime, proceed as follows:

1. Open the task configuration via **WinStudio** ► **Tools** ► **Status**. The "Execution Tasks" dialog is displayed.
2. In the list of tasks, select **OPC UA Client Runtime**.
3. Click on **Start**.
The "Start" dialog is displayed.
4. Select **Automatic** and click on **OK**.
5. Click on **OK** to close the "Execution Tasks" dialog.

11.5 Configuring a TCP/IP Client Connection to Another Project

TCP/IP client task/worksheets are used to communicate with different "WinStudio" project servers.

TCP/IP client and server tasks enable two or more projects to keep their databases synchronized.

These tasks use the TCP/IP protocol to provide communication between projects.

Before using the TCP/IP client and server tasks, it has to be ensured that the TCP/IP (ethernet) communication is properly configured and running on both servers.

- **To configure the server, proceed as follows:** On the server itself, nothing has to be configured. Only the "WinStudio" TCP/IP Server task has to be executed. It can either be executed automatically on starting the runtime or manually in the task configuration (either local or by remote access). After running this program, a small icon is displayed in the system tray.
- To close the "WinStudio" TCP/IP Server task, proceed as follows: Right-click on the **TCP/IP Server** icon in the system tray and click **Exit**.
- **To configure the client, proceed as follows:** Use "TCP/IP worksheets" to specify the server IP address and the tags to be used with the server on the client system. These may be created in the following folder: **WinStudio** ► **Tags** ► **Communication** ► **TCP/IP**

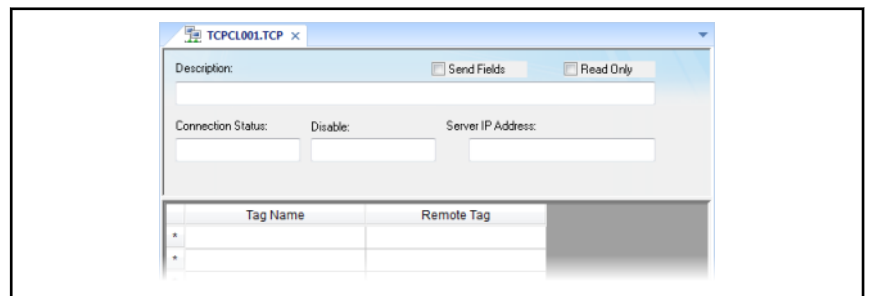


Fig. 11-15: TCP/IP driver sheet configuration

Use the following parameters to complete the TCP/IP client configuration:

- **Description:** Enter a description of the TCP/IP driver sheet for documentation purposes. The TCP/IP client task ignores this information.
- **Connection Status:** Enter a tag name to be updated according to its connection status by the TCP/IP client configuration task. A tag value of zero indicates that the connection is okay. Any other tag value indicates an error code returned by the "Windows Socket" library.
- **Disable:** Enter a tag name in this field. If this tag has any value other than 0, this TCP/IP worksheet is disabled. If this field is used, the TCP/IP client worksheet can be activated/deactivated during runtime.

Communication with Other Devices

- **Server IP Address:** Enter the IP address and Port (optional) of the target server, for example, "169.254.182.158:123". The Port is to be the same on both the client and server stations.

String tags can be entered in curly brackets (e.g., {tagname}) to dynamically change this address during runtime.

- **Send Fields:**
 - If this checkbox is "disabled", TCP/IP client/server tasks "only" exchange tag values and their TimeStamp and Quality.
 - If this checkbox is "enabled", TCP/IP client/server tasks "also" exchange the Min, Max, Ack, Unit, LoLoLimit, LoLimit, HiLimit, HiHiLimit, RateLimit, DevSetPoint, DevPLimit, and DevenLimit tag fields.



Additional fields can be added to the TCP/IP communication or fields can be disabled individually. For further information, contact the distributor.

- **Read Only:** If this option is enabled, the entire communication takes place in one direction and no tag values are returned to the specified server. This is useful if the TCP/IP client is only used to read data from other projects. Enabling this option may considerably improve safety and stability during runtime.
- **Tag Name:** Enter the tags to be shared with the server. If the tag is an array or class (or both), the project automatically activates all array and class elements for TCP/IP Communication by default. To configure a specific array and/or class element, enter the array index or class element in square brackets followed by the tag name. For example, "level[3].member".
- **Remote Tag** ("optional"): Enter the name of a tag to be linked with the tag specified in the **Tag Name** field. If this field is left blank, the project uses the same tag name used in the client and in the server.



If an array is to be shared, the tag in the server is to contain the same number of elements as the tag in the client. If the tag is a class, the class definition is to be the same in both server and client programs. If these rules are not adhered to, unexpected results can occur.

The TCP/IP client task can either be executed automatically on startup of the runtime environment or manually in the task configuration (**WinStudio ► Tools ► Status**). After starting the TCP/IP task, a small icon is displayed in the system tray.

The parameter "ConnectionRetryTimeout" is only used by the client.

11.6 Configuring a DDE Client Connection to a DDE Server

DDE worksheets are used to configure DDE client connections to DDE server applications such as Microsoft Excel (or any other Windows program supporting this interface).

Dynamic Data Exchange (DDE) is a protocol for dynamic data exchange between Windows applications, such as Excel. DDE communication is an interaction between server and client programs. "WinStudio" provides interfaces that run as clients or as servers. Refer to DDE client runtime and DDE server in the Runtime tasks (refer to **Tools ► Status** in the context menu of the "WinStudio" folder).

Communication with Other Devices

- To run as a server, start the DDE or NetDDE server task as described in "Runtime tasks".
- To execute as client, the DDE driver sheet is configured in the **WinStudio ► Tags ► Communication ► DDE** folder.

Network Dynamic Data Exchange (NetDDE) is an extension of DDE that works across computers on a network.

- To run "WinStudio" as server for a NetDDE connection, start the DDE server application.
- To run "WinStudio" as a client for a NetDDE connection, use the same DDE driver sheets with the respective configuration to address a "WinStudio" project.



When running NetDDE, "WinStudio" accepts only WRITE triggers. To read data, a write command has to be configured on the server computer.

To create a new "DDE" worksheet, select "New Driver Sheet" in the context menu of the "DDE" folder. Afterwards, double-click on the new element to open the driver sheet editor.

Fig. 11-16: DDE worksheet

The "DDE" worksheet dialog is divided into two areas:

- "Header" area (top section) containing information for the whole group and defining the tags to start the reading and writing process and to receive the connection status.
- "Body" area (bottom section), to define tags in the project and items related to the DDE server application.

Every DDE interface is based on addressing an application using the following three structures:

- Project name
- Topic
- Item

This information has to be determined before configuring the DDE driver sheet in the DDE server application.

Use the parameters in the DDE client worksheet header area as follows:

- **Description:** Enter a description of the DDE worksheet for documentation purposes.
- **Project Name:** Enter the DDE server application name.
- **Topic:** Specify a topic in the server application.
- **Connect:** Enter a tag to control the connection of the "WinStudio" DDE client and DDE server application. If this tag is set to 1, a connection to

Communication with Other Devices

the server is requested. If the connection cannot be established or fails, "WinStudio" resets the tag to 0. If the connection is OK, this value remains set to 1.

- **Read Trigger:** Enter a tag as read command for the table. If this tag changes its value, "WinStudio" calls the variable values from the DDE server. This parameter can only be used with local DDE, not with NetDDE servers.
- **Enable Read when Idle:** Enter a tag value higher than zero to enable reading of the equipment.
- **Read Status:** Contains the status of the read command.
- **Write Trigger:** Enter a tag triggering the write command.
- **Enable Write on Tag Change:** Enter a tag value higher than zero to enable the communication driver to check continuously for changes in a tag value in the worksheet. If the driver detects a change, it writes the changed tag to the device, along with the tag address.
- **Write Status:** Contains the status of the writing command.

Use the DDE client body area parameters as follows:

- **Tag Name:** Enter a tag to read or write the "WinStudio" database from the DDE server application.
- **Item:** Enter the ITEM part of the DDE address on the server. Refer to the server software documentation for information about the proper syntax for **APP**, **TOPIC**, and **ITEM**.

The **Topic** and **Item** fields can be configured with tags in the address according to the following syntax:

text{tag}

"WinStudio" evaluates the value of {tag} in a String and uses it in the address. For example:

- Topic: topic_{tag_topic_name}_example
- Item: {tag_item_name} or
A{tag_number}

Configuring a NetDDE connection is similar to configuring a DDE connection, except for the "Name" and "Topic" in the header of the application. Before starting the tests, verify that the DDE server on the station with which data is to be exchanged is enabled.



When connecting to servers other than "WinStudio", the proper syntax for **APP**, **TOPIC** and **ITEM** can be found in the server documentation.

To define the tags to initiate reading and writing as well as tags that receive the connection status, use the NetDDE client worksheet header parameters as follows:

- **Project Name:** Enter **computer_name\NDDE\$**, where **computer_name** has to be a valid network computer name.
- **Topic:** Enter the topic name **UNISOFT\$** to connect to another "WinStudio" station.

Use the DDE client worksheet body parameters to link each tag to each ITEM part of the DDE server address, as follows:

- **Tag Name:** Specify the "WinStudio" local database tag name that is related to a remote tag name.

Communication with Other Devices

- **Item:** Specify the remote tag name that is related to the local tag name.



By default, the DDE client module supports DDE servers that handle String data in the UNICODE format. If the DDE server handles String data in the ASCII ANSI format, the following settings have to be configured manually in the **project_name.app** file (Notepad can be used to edit this file):

[Options]

DDEANSI=1

12 Project Security

12.1 Project Security, General Information

WinStudio includes a project security system that manages how users and user groups can access projects during development and runtime.



The WinStudio security system is only available if the **Application Type** "WinStudio Application" is selected in the properties of the visualization device. For other application types, functions are protected during runtime by the IndraWorks function "Access Protection".

12.2 Security Modes

In addition to managing users and groups locally, entirely within a single project, pre-defined users and groups from other "WinStudio" projects or an LDAP-compliant domain server can also be applied.

"WinStudio" supports four security modes:

Local Only

This is the standard mode for most projects: Users and groups are created in the development environment and apply only to the project for which they are created.

Distributed – Server

This mode is similar to **Local Only**, except that the project's security system configuration is also made available to other "WinStudio" projects (that are set to **Distributed – Client**) on the same network.

Additionally, if the project loses its security system configuration for some reason, it can be reimported from one of its client projects.

Distributed – Client

If this mode is selected, the project receives its entire security system configuration from another "WinStudio" project (that is set to **Distributed – Server**) on the same network.

The project saves this configuration and can continue to run even if communication to the server project is lost.

Domain (LDAP)

The Lightweight Directory Access Protocol (LDAP) is a recognized standard for managing users and groups across many different applications on a network.

If this mode is selected, the project receives its users and groups from an LDAP-compliant domain server, such as Microsoft Active Directory for Windows or OpenLDAP for Linux. However, only user names, passwords, and group memberships are taken from the domain. Specific rights still have to be configured within the project.

12.3 Security Access Levels

Almost all items of a project – screen objects, object animations, project screens, task worksheets – can be assigned a security access level. This access level determines the user groups that can edit the item during development and/or use it during runtime.

There are 255 possible access levels, allowing fine grading. Each user group is configured with ranges of levels for both development and runtime, and the ranges of the group may overlap.

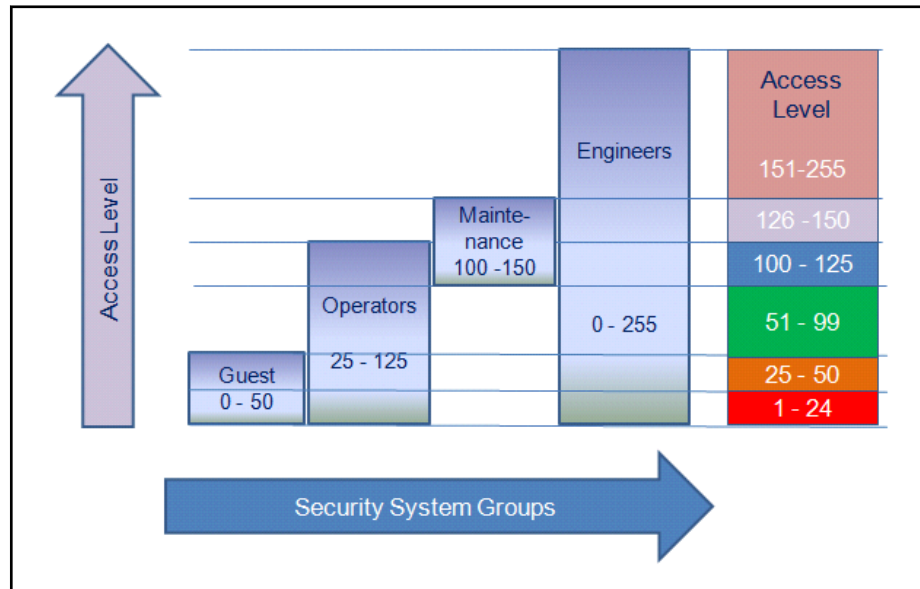


Fig. 12-1: Example of security access levels

This means that for a user to be able to edit and/or use an item, the item's access level has to fall within the range specified for that user's group.

For example: **UserA** of **GroupA** has a security access level range of "1-10" and **UserB** of **GroupB** has a security access level range of "5-15". Additionally, the project contains the following items:

- Item #1 has Access Level = 1
- Item #2 has Access Level = 7
- Item #3 has Access Level = 12
- Item #4 has Access Level = 20

This results in the following access levels for the items:

- Only **UserA** can access Item #1
- Both users can access Item #2
- Only **UserB** can access Item #3
- Neither user can access Item #4



The default access level for all items is 0 and all users can use all items at that level.

Setting access levels for different types of items

The access levels for WinStudio screens can be set in the [Screen Attributes dialog on page 156](#). Security (access level) settings in the "Screen Attributes" dialog

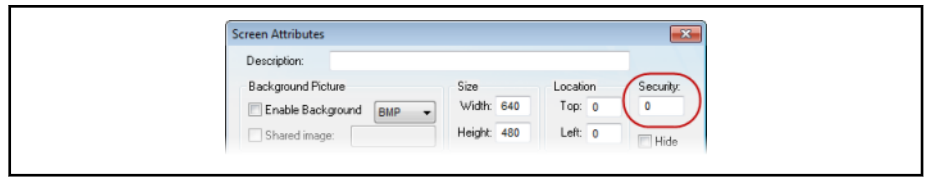


Fig. 12-2: Security (access level) settings in the Screen Attributes dialog

For screen objects and object animations, the access level can be set in the individual [Object Properties dialog on page 164](#). (If the option is not available in the main dialog, it is generally available in one of the sub-dialogs.) Security (access level) settings in the "Object Properties" dialog

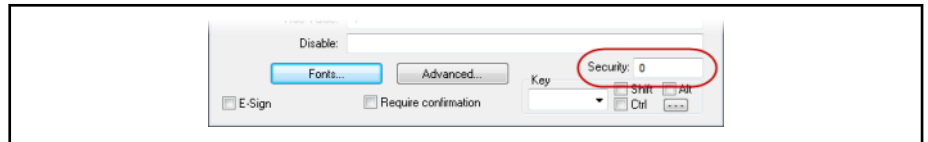


Fig. 12-3: Security (access level) settings in Object Properties dialog

For task worksheets, do the following:

1. Open the worksheet for editing.
2. Click anywhere in the body of the worksheet. The entry **WinStudio <worksheet> ► Change security level** in the main menu of IndraWorks is enabled.
3. Click on **Change Security Level**. The "Security" dialog is displayed.

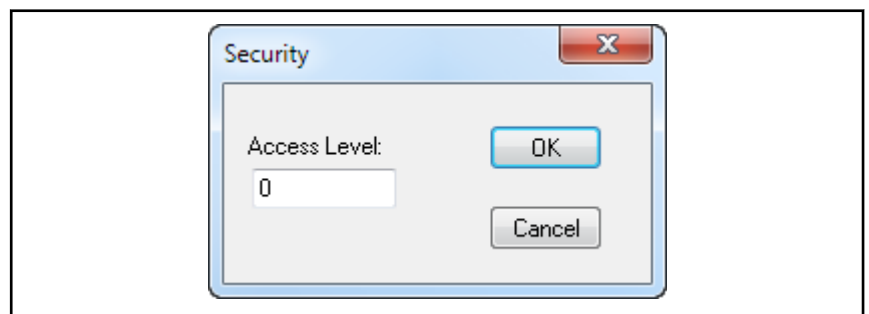


Fig. 12-4: Security dialog

4. In the **Access Level** field, enter an access level for editing the worksheet.
5. Click on OK.

12.4 Using the Security System Configuration Wizard

The Security System Configuration Wizard guides through the steps of configuring the project security system. To start the configuration wizard, select **Tools ► Safety settings** in the context menu of the **WinStudio** folder.

Project Security

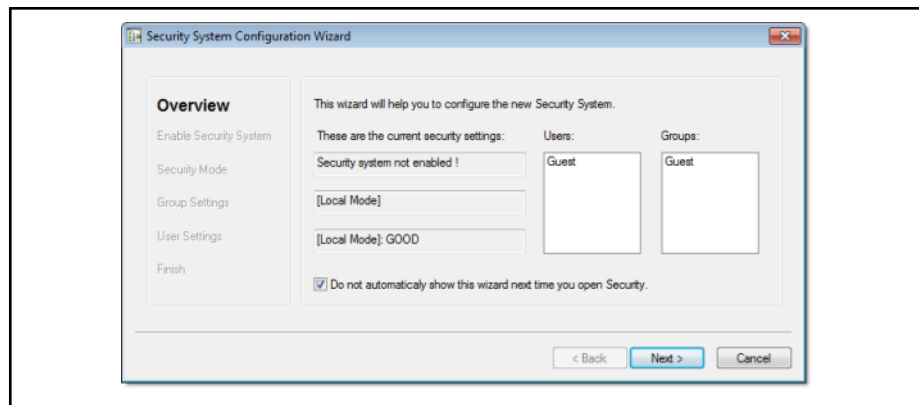


Fig. 12-5: Security System Configuration Wizard

1. The first page of the wizard is displayed. This page always shows how the security system is currently configured.
2. Click on **Next**. The second page of the wizard is displayed.

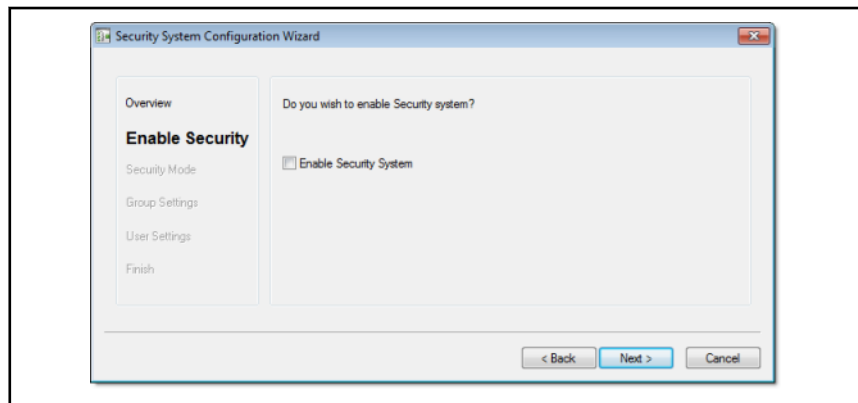


Fig. 12-6: Security System Configuration Wizard (2)

Select **Enable Security System**.

3. Click on **Next**. The third page of the wizard is displayed.

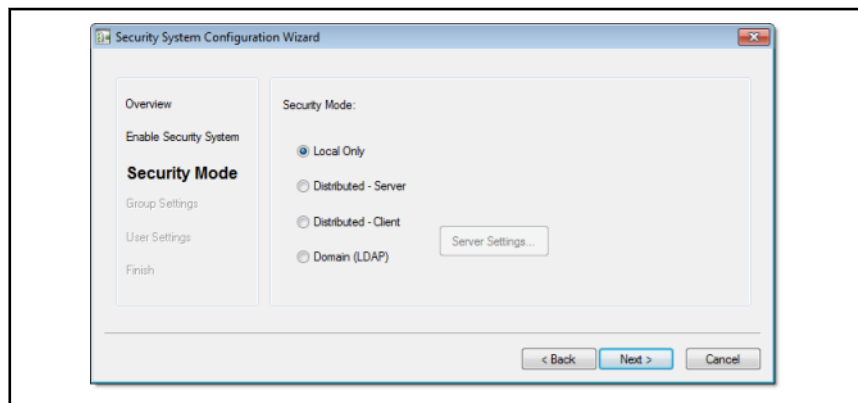


Fig. 12-7: Security System Configuration Wizard (3)

Select the [Security Mode](#) on page 357.

For **Distributed – Client** and **Domain (LDAP)**, click **Server Settings** and configure the [Required settings](#) on page 362.

4. Click on **Next**. The fourth page of the wizard is displayed.

Project Security

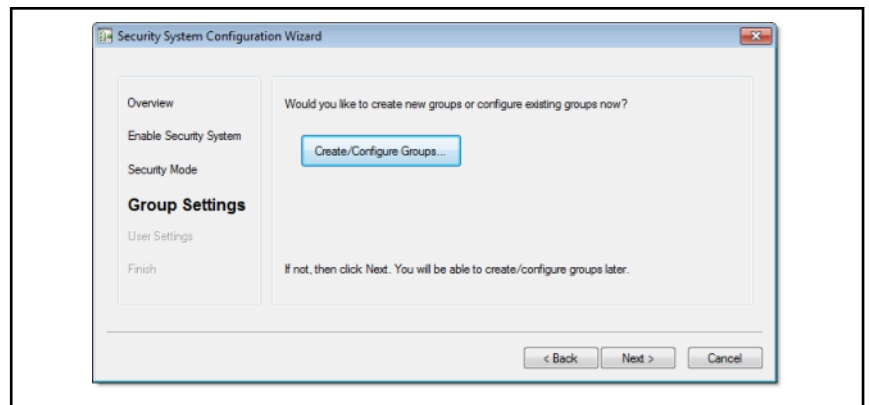


Fig. 12-8: Security System Configuration Wizard (4)

If groups have to be created or configured, click on **Create/Configure Groups**. The [Group Account dialog on page 370](#) is displayed. If this dialog is closed, the system automatically returns to the wizard.

5. Click on **Next**. The fifth page of the wizard is displayed.

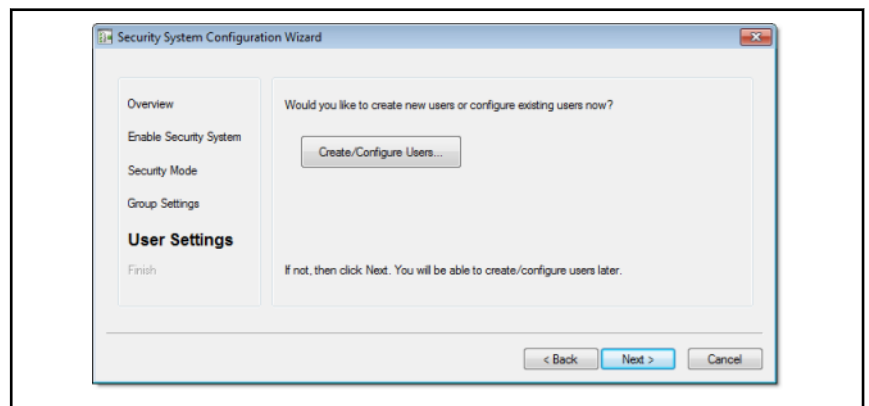


Fig. 12-9: Security System Configuration Wizard (5)

If users have to be created or configured, click on **Create/Configure Users**. The [User Account dialog on page 374](#) is displayed. If this dialog is closed, the system automatically returns to the wizard.

6. Click on **Next**. The sixth page of the wizard is displayed.

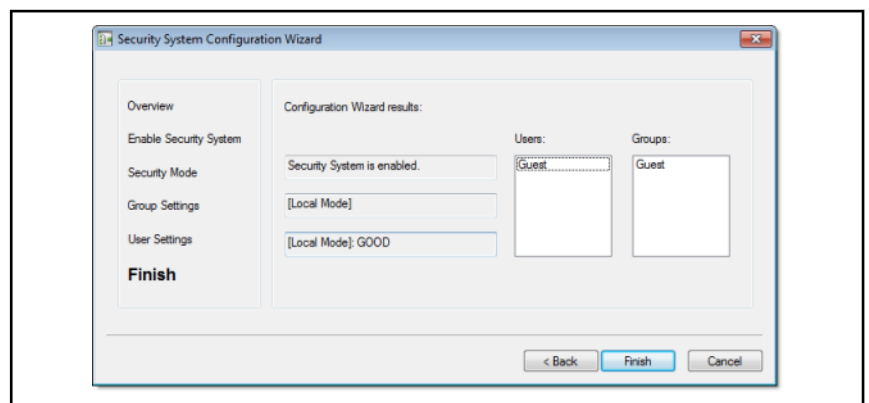


Fig. 12-10: Security System Configuration Wizard (6)

7. Review the configuration and click **Finish** to close the wizard.

Project Security

12.5 Configuring Server Settings for Security Modes

12.5.1 Configuring Server Settings for Security Modes, General Information

If the security mode of the project is either set to Distributed – Client or Domain (LDAP), the server settings have to be configured as well.

Click **Server Settings** in the [Security System Configuration Wizard on page 359](#) or the [Security System main dialog on page 375](#) to open the respective "Server Settings" dialog.

Configuring the server settings for the security mode: Distributed – Client

To configure the server settings in the Server Settings dialog for the security mode, proceed as follows:

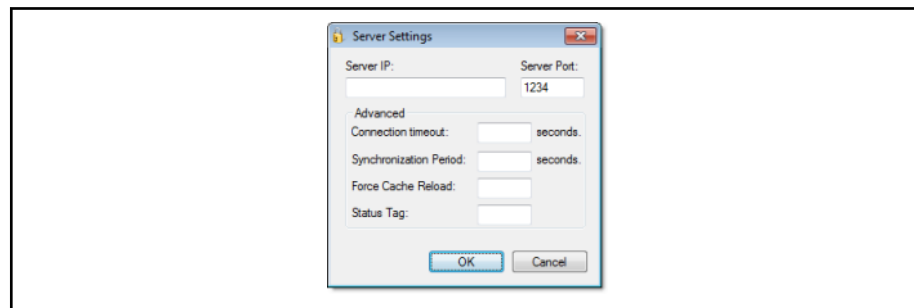


Fig. 12-11: Distributed – Client

1. In the **Server IP** and **Server Port** fields, enter the IP address of a runtime project that is set to **Distributed – Server**.
2. In the **Connection timeout** field, enter the timeout (in seconds) after which the client is to attempt to reconnect to the server (a typical Connection timeout is 3 seconds).
3. In the **Synchronization Period** field, enter the frequency (in seconds) at which the client is to synchronize its security system configuration with the server configuration. (A typical Synchronization Period is 10 seconds.)
4. In the **Force Cache Reload** field, enter the name of an application variable (Integer or Boolean type). If the tag value is TRUE (not zero) and the specified server has a timestamp older than the client, the local security system is updated with outdated server information.
5. In the **Status Tag** field, enter the name of an application tag (Integer type) that is to receive a value indicating the status of the server connection. While the project is running, the following values are applicable: For further information, refer to [GetSecuritySystemStatus on page 538](#).

Value	Description
0	No cache
1	Updated cache
2	Outdated local cache
3	Outdated server cache
4	Disconnected from server

6. Click on **OK**.

Configuring the server settings for the security mode: Domain (LDAP)

To configure the server settings in the Server Settings dialog for the security mode, proceed as follows:

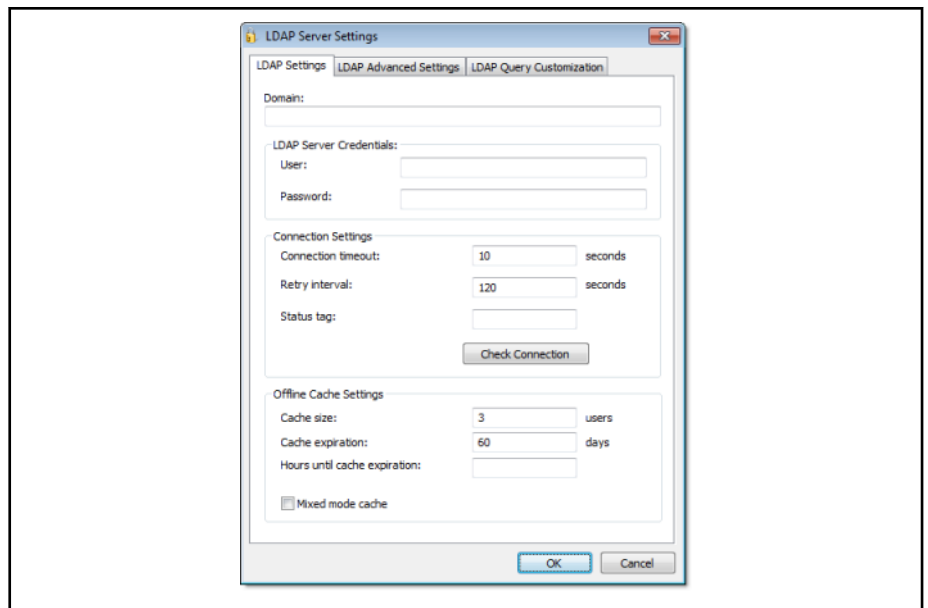


Fig. 12-12: Domain (LDAP)

1. In the **Domain** field, enter the domain name of the LDAP server.
2. In the **User** and **Password** fields, enter the access data for the LDAP server. To access group and user lists, sufficient user rights are required. Please contact the responsible LDAP administrator.
3. In the **Connection timeout** field, enter the timeout (in seconds) after which the client is to attempt to reconnect to the LDAP server (a typical connection timeout is 4-5 seconds).
4. In the **Retry interval** field, enter the frequency (in seconds) at which the client is to try to connect to the LDAP server if the connection could not be established.
5. In the **Status Tag** field, enter the name of an application tag (Integer type) that is to receive a value indicating the status of the server connection. While the project is running, the following values are applicable:

Value	Description
0	Connection timeout
1	Binding timeout
2	Request timeout
3	Disconnected
4	Connected
5	Query returned no users or groups
6	Invalid user or group

For further information, refer to [GetSecuritySystemStatus](#) on page 538.

6. Click **Check Connection** to confirm that the project can connect to the specified domain. If this is not possible, review and correct the settings.
7. Click on **OK**.

Project Security



The project security system caches the most recent users in case the project loses its connection to the LDAP server. These users are still able to log onto the project. The cache size can be unlimited by setting **Cache size** to "0" and never expires by setting **Cache expiration** to "0".

To monitor the cache during project runtime, enter the name of a project tag in the **Hours until cache expiration** field. This tag receives a value equal to the number of hours until the cache expires.

If users are to be enabled to log on with either local or server-defined accounts, select **Mixed mode cache**. Local users and groups can only be created if the **Security Mode** is set to **Local Only**. After creation, they can be kept in the cache if the **Security Mode** is set to **Domain (LDAP)**.

In the "LDAP Advanced Settings" tab of the dialog, there are additional settings that are to be configured only by experienced LDAP administrators:

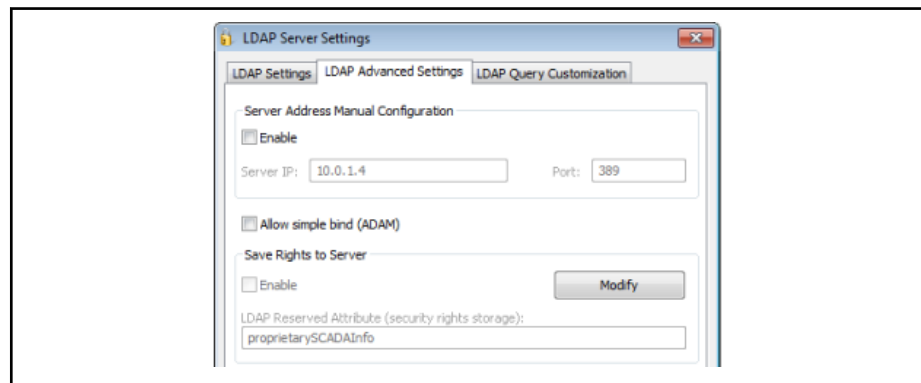


Fig. 12-13: LDAP Advanced Settings

Server Address Manual Configuration

If for some reason the LDAP server cannot be accessed using its domain name, the server IP address can be manually configured: Select **Enable** and enter the IP address and port number of the LDAP server.



The default port for LDAP is 389. However, please verify the port number with the server administrator.

Allow simple bind (ADAM)

LDAP normally requires secure binding for authentication, but Active Directory Application Mode (ADAM) in Windows Server 2003 does not fully support secure binding. To allow simple binding with an ADAM server, select this option.



Simple binding means that user data is sent in plain text. For this reason, the connection is to be secured by other means such as VPN, TLS/SSL or proxies.

Save Rights to Server

By default, "WinStudio" security rights are saved entirely within the project. However, those rights can be saved back to the LDAP server, either to make them available to other projects that use the same LDAP server or for simple redundancy.

To make this option work, the server LDAP schema is to be extended by additional information about the project security

system. Refer to [Extending the LDAP schema to allow saving of security rights on page 365](#).

Afterwards, click on **Modify** to provide the LDAP server credentials and select **Enable**.

In the "LDAP Query Customization" tab of the dialog, the formatting of LDAP server queries can be further customized:

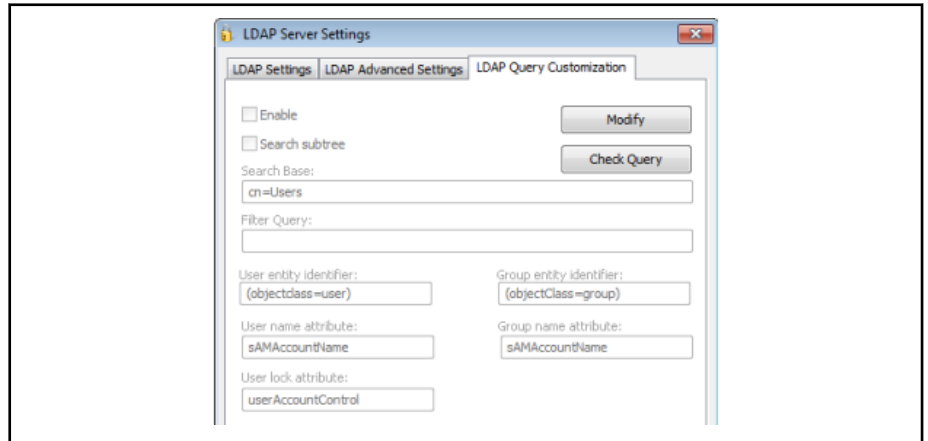


Fig. 12-14: LDAP Query Customization

By default, the LDAP server provides a list of "all" registered users and groups. In a large or complex network environment, this may lead to an impractically long list for project security system configuration. To restrict the list of users and groups, the LDAP query can be customized to eliminate any users and groups that never are to access the project: Click on **Modify** to enter the LDAP server credentials, select **Enable**, and then configure the **Search Base** and **Filter Query** settings. For the proper syntax, consult the LDAP server documentation.

Also, some non-standard LDAP implementations – such as Linux-based LDAP servers and Active Directory Application Mode (ADAM) in Windows Server 2003 – use different entity identifiers and attributes. These can be customized in this dialog, however, only by experienced LDAP administrators.

Example of alternative attributes in ADAM

LDAP server	User name attribute	Group name attribute	User lock attribute
Active Directory	sAMAccountName	sAMAccountName	userAccountControl
Active Directory Application Mode (ADAM)	Name	Name	userAccountControl

12.5.2 Extending the LDAP Schema to Allow Saving of Security Rights

In order to save "WinStudio" project security rights back to a Domain (LDAP) server, the server LDAP scheme has to be extended for these additional information.

The server has to already be configured and running on the network, and sufficient access rights are required to change to the server configuration.

In this procedure, new attributes with the designation "proprietarySCADAInfo" are created to include the "WinStudio" project security rights. Afterwards, the attributes are assigned to the classes "user" and "group" in the server configuration. These classes correspond to users and groups in the project security system.

Project Security

Please note that this procedure only illustrates how the scheme is to be extended in Microsoft Active Directory under Windows Server 2003. For different LDAP servers and operating systems, a different procedure applies, however, the basic steps are generally identical. Please refer to the LDAP server documentation.

NOTICE

Extending the LDAP scheme of a server cannot be undone.

1. Register the schema management DLL.

- 1.1 Click on **Start ► All Programs ► Accessories ► Command Prompt**.

A command prompt window is displayed.

- 1.2 In the prompt, enter **cd %SystemRoot%\System32** and press Return.

The working directory is changed.

- 1.3 Enter **regsvr32 schmmgmt.dll** and press Return.

If the DLL is successfully registered, a confirmation message is displayed.

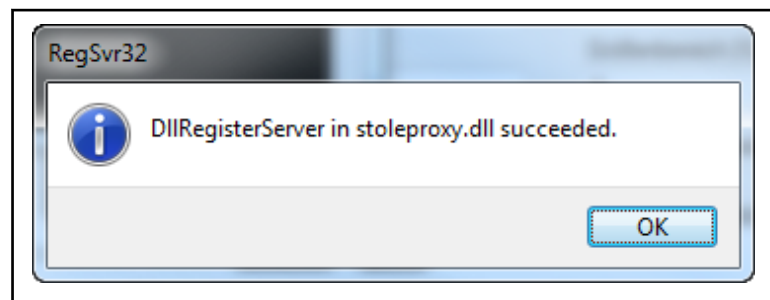


Fig. 12-15: Confirmation message

- 1.4 Click OK to close the message.

- 1.5 Close the command prompt window.

2. Add the Active Directory Schema snap-in to the console root.

- 2.1 Click on **Start ► All Programs ► Accessories ► Run**. A Run dialog is displayed.

- 2.2 In the **Open** field, enter "mmc" and click on **OK**.

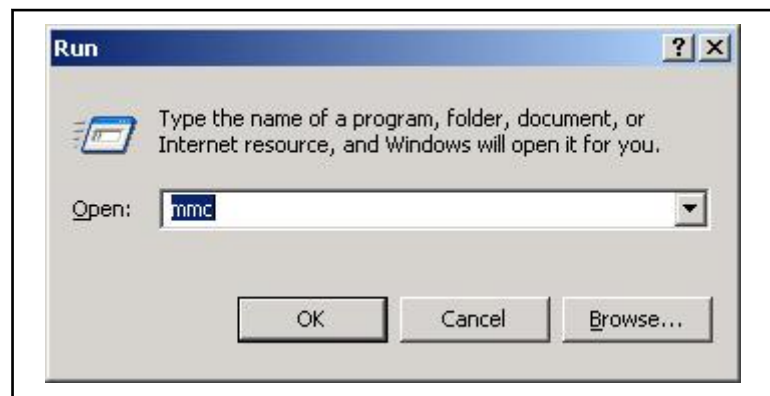


Fig. 12-16: Open field

Project Security

(If User Access Control (UAC) is enabled, carrying out changes to the Microsoft Management Console is to be confirmed. Click on Yes.) A console window is displayed.

- 2.3 In the console window, click on **File ► Add/Remove Snap-in**.

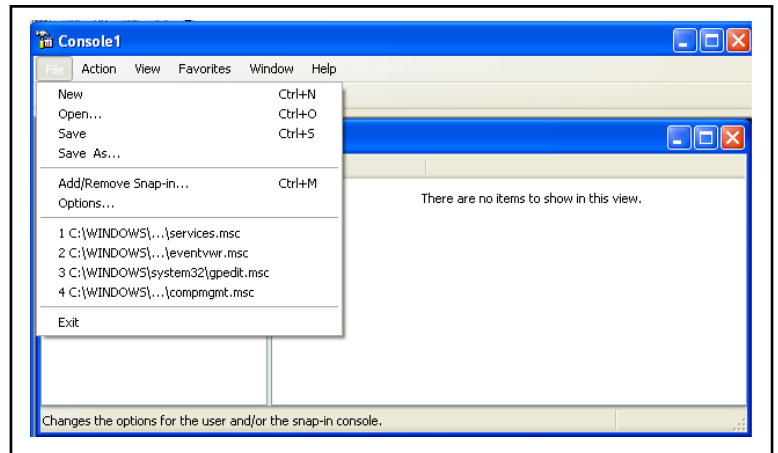


Fig. 12-17: Menu: Add/Remove Snap-in

The Add/Remove Snap-in dialog is displayed.

- 2.4 In the **Snap-ins added to** list, select **Console Root** and click on **Add**. The "Add Standalone Snap-in" dialog is displayed.

- 2.5 In the list of available snap-ins, select **Active Directory Schema** and click on **Add**.

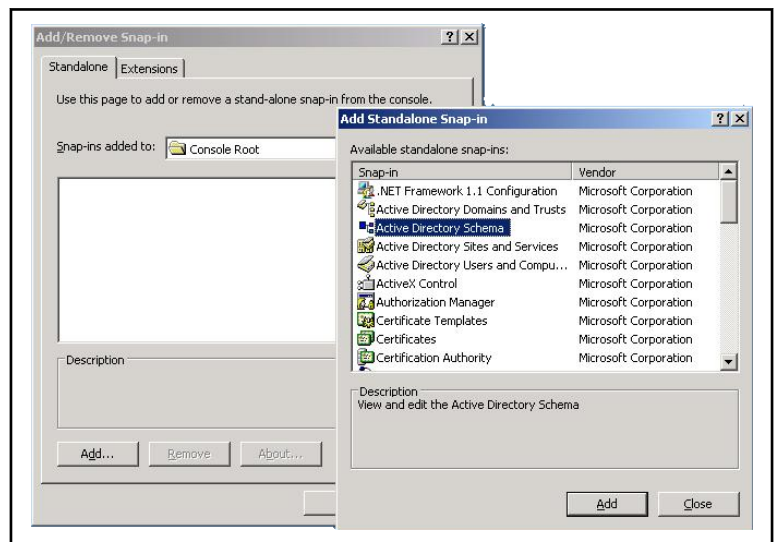


Fig. 12-18: Dialog Add Standalone Snap-in

The snap-in is added to the Console Root.

- 2.6 Click **OK** to close the Add/Remove Snap-in dialog.
3. Create the proprietary SCADAInfo attribute in the Active Directory Schema snap-in The proprietary SCADAInfo attribute is added to the list.
- 3.1 In the **Console Root** tree-view, expand **Active Directory Schema**.

Project Security

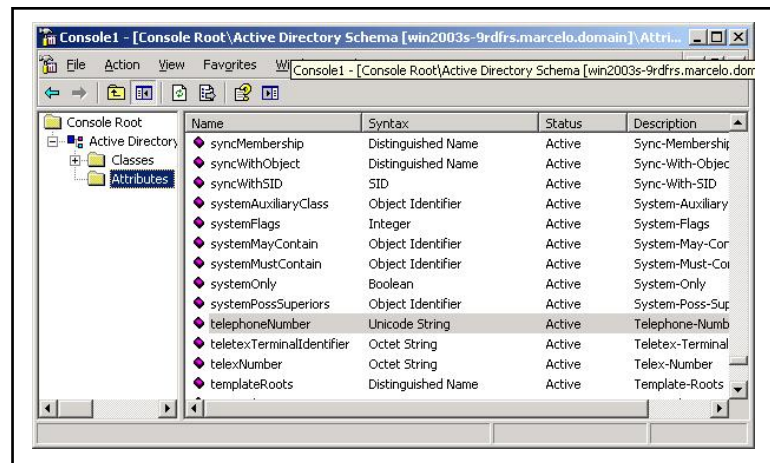


Fig. 12-19: Active Directory Schema

- 3.2 Right-click on Active Directory **Schema** ► **Attributes** and click on **New Attribute** in the context menu. A message is displayed explaining that the schema is permanently changed.
- 3.3 Click on Continue. A Create New Attribute dialog is displayed.
- 3.4 In the dialog, complete the fields as follows.
 - **Full Name:** proprietarySCADAInfo
 - **LDAP Display Name:** proprietarySCADAInfo
 - **Unique X500 Object ID:** 0.7.7777.77777777.777.7.7



A unique Object ID is to be used.

Description: proprietarySCADAInfo

Syntax: Octet String

Minimum: 0

Maximum: 10240

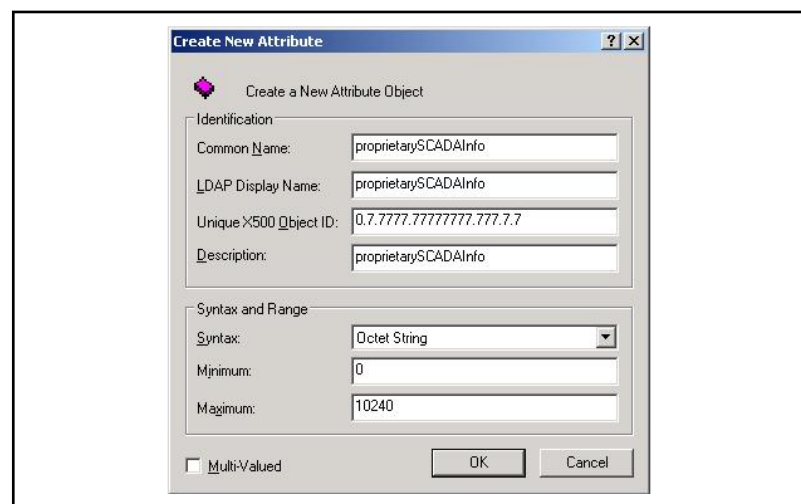


Fig. 12-20: Dialog: Create New Attribute

- 3.5 Click on **OK** to close the dialog.
- The proprietarySCADAInfo attribute is added to the list.

Project Security

4. Add the proprietarySCADAInfo attribute to the **person** and **group** classes.

- 4.1 In the **Console Root** tree-view, select **Active Directory**. **Schema** ► **Classes**.

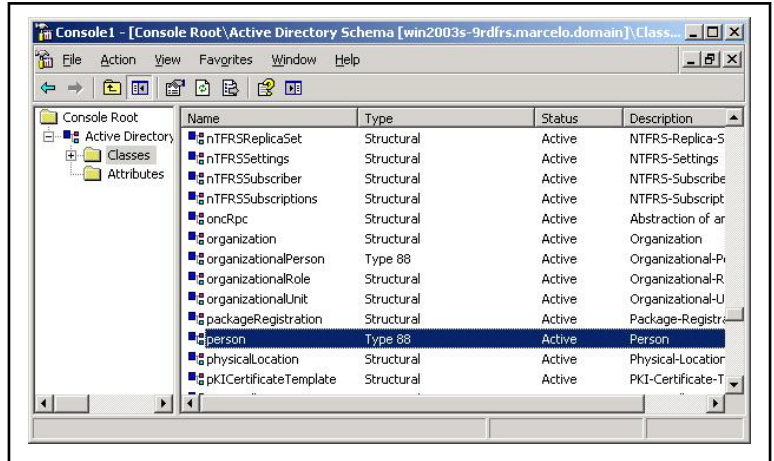


Fig. 12-21: Dialog Active Directory Schema > Classes

- 4.2 In the list of classes, right-click on **person** and click on **Properties** in the context menu.

The Properties dialog is displayed.

- 4.3 In the dialog, click on the **Attributes** tab.

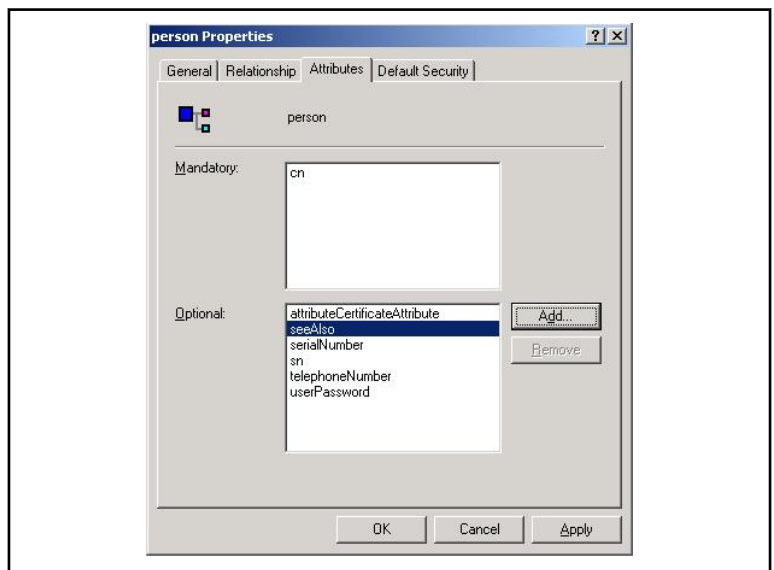


Fig. 12-22: Dialog person Properties, Attributes

- 4.4 Click on **Add**. The Select Schema Object dialog is displayed.
- 4.5 In the list of schema objects, select **proprietarySCADAInfo** and click on **OK**.

Project Security

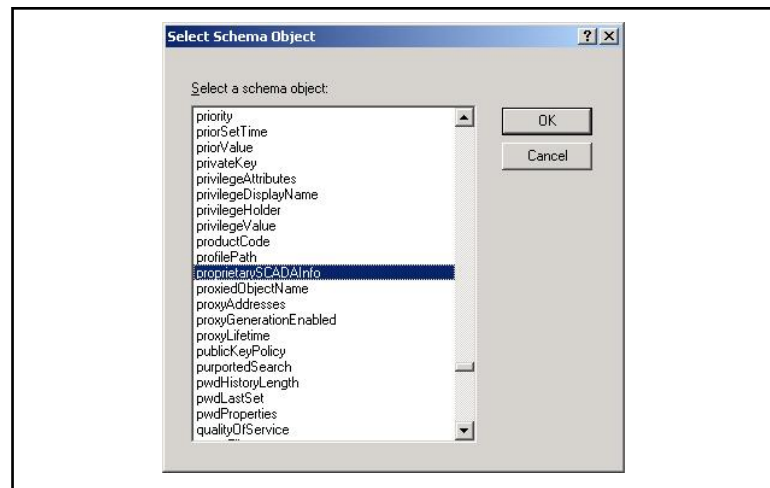


Fig. 12-23: Select Schema Object dialog

The attribute is added to the class properties.

- 4.6 Click on **OK** to close the Properties dialog.
- 4.7 Repeat steps b through f for the **group** class.
5. In the **Console Root** tree-view, right-click on **Active Directory Schema** and select **Refresh** in the context menu.
6. Click on **File ► Close** to close the console window.
7. Restart the server.

12.6 Creating and Configuring Groups

Use the "Group Account" dialog to create and delete user groups and to configure the access rights of a selected group.

Accessing the dialog

Assuming the project security system has already been enabled (i.e., the Security System Configuration Wizard has been used at least once), this dialog can be accessed as follows:

- Open the [Security System dialog on page 375](#) and click on **Groups**; or
- In the WinStudio context menu, select the entry **Tools ► Add Group** and click on **Groups**

The dialog in detail

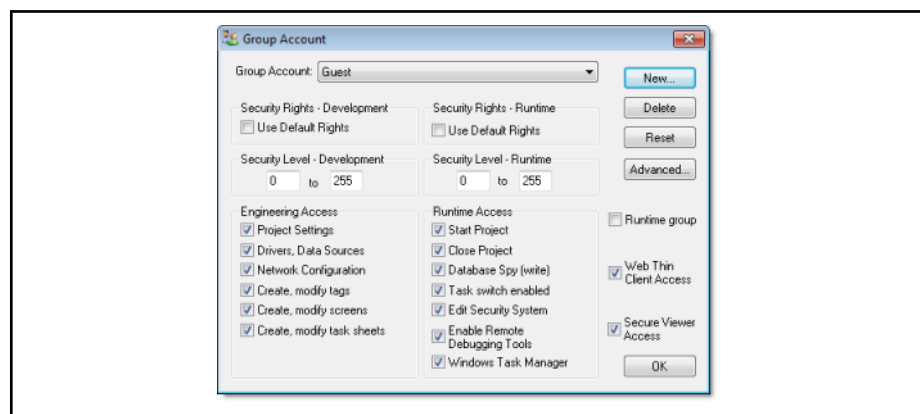


Fig. 12-24: Group Account dialog

Project Security

Area / Element		Description
Group Account		The user group that is currently configured. Note: There are two default groups for all projects: Guest and (Default Rights). Note: If the security mode is set to Domain (LDAP), please note that the built-in groups in Microsoft Active Directory are not displayed in this list of groups and cannot be added to the project.
New		Creates a new group. In the "New Group Account" dialog, enter the name of the new group and then click OK .
Delete		Deletes the currently selected group.
Reset		Resets the privileges of the currently selected group to match the (Default Rights) group. This does not lock the group to the default; further changes can be carried out. To lock the group, see Use Default Rights below.
Advanced settings		Opens the "Group Account Advanced" dialog (see below).
Security Rights – Development	Use Default Rights	Locks the development privileges of the currently selected group to those configured for the (Default Rights) group. If changes are made to the (Default Rights) group, then they also apply to this group.
Security Level – Development	Range	The range of Access Levels on page 358 that this group can access in the development application.
Engineering Access	Project Settings	Members of the group may modify the Project Settings on page 62 .
	Drivers, Data Sources	Members of the group may create and modify device drivers and external data sources on page 333 .
	Network Configuration	Members of the group may create and modify TCP/IP Client worksheets on page 351 .
	Create, modify tags	Members of the group may create and modify project tags on page 106 .
	Create, modify screens	Members of the group may create and modify project screens on page 155 .
	Create, modify task sheets	Members of the group may create and modify task worksheets on page 315 .
Security Rights – Runtime	Use Default Rights	Locks the runtime privileges of the currently selected group to those configured for the (Default Rights) group. If changes are made to the (Default Rights) group, then they also apply to this group.
Security Level – Runtime	Range	The range of Access Levels on page 358 that this group can access in the runtime project.

Project Security

Area / Element		Description
Runtime Access	Start Project	Members of the group may run the project.
	Close Project	Members of the group may stop the project.
	Database Spy (write)	Members of the group may write values to the project database using the Database Spy window on page 84 . Note: This only applies to projects running locally. For projects running remotely, see Enable Remote Debugging Tools below.
	Task switch enabled	Members of the group may switch away from the runtime project to another Windows task. (This option does not apply to projects running on Windows Embedded target systems; the user may always switch away from the runtime project.)
	Edit Security System	Members of the group may edit the project security system during runtime. Note: This option may not be deactivated for the own group as, in this case, own changes cannot be undone.
	Enable Remote Debugging Tools	Members of the group may use the Remote Database Spy on page 388 and the Remote LogWin on page 389 .
Windows Task Manager		Members of the group may open the Windows Task Manager. (This option does not apply to projects running on Windows Embedded target systems; the user may always open the Windows Task Manager.) Note: Clearing this option means disabling the Task Manager during runtime, which requires Administrator rights. The project has to be executed with higher rights.
Runtime group		A user created during runtime by means of the CreateUser on page 536 function may be assigned to this group.
Web Thin Client Access		Members of the group may access the runtime project by using a Web Thin Client.
Secure Viewer Access		Members of the group may access the runtime project by using a Secure Viewer.

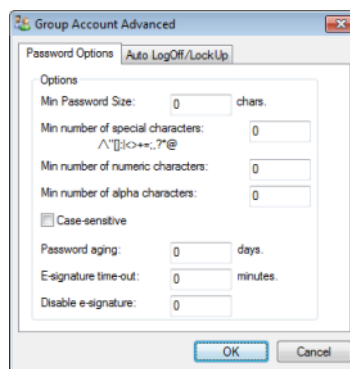


Fig. 12-25: Advanced dialog – Password Options

Project Security

Area / Element	Description
Min Password Size	To make user passwords more complex and therefore more secure, passwords can be required to contain a certain number of alpha (A-Z, a-z), numeric (0-9), and special (punctuation) characters. If users are prompted to change the password, for example, if the old password expires (refer to Password aging below), the new password is not accepted unless it meets these requirements.
Min number of special characters	
Min number of numeric characters	
Min number of alpha characters	
Case-sensitive	If this option is enabled, passwords are case sensitive, i.e., passwords created with both upper and lowercase characters "have to" be entered the same way by the user. Note: In projects created with "Rexroth WinStudio v6.5" to "v7.2+SP1", all passwords are case sensitive.
Password aging	Validity (in days) of the password for all users in this group. After the specified number of days, the project forces users to change their password: If users try to log in, the "Change Password" dialog is automatically displayed and the logon process cannot be completed before a new password is specified. By default, users have to choose a new password that is different from the old password. To change this and to enable users to re-use the same password, manually edit the project file (project_name.app) to include the following setting: [Security] ChangePasswordMode=1 To make passwords never expire, set Password aging to "0".
E-signature time-out	Timeout period (in minutes) of the E-sign prompt for all users in this group. Users have to enter their user name and password before the specified timeout to use project functions that require an e-signature.
Disable e-signature	If the value in this box is TRUE (not zero), users in this group cannot use any project functions that require an e-signature. In this field, project tags can be configured to dynamically enable/disable the e-signature during runtime.

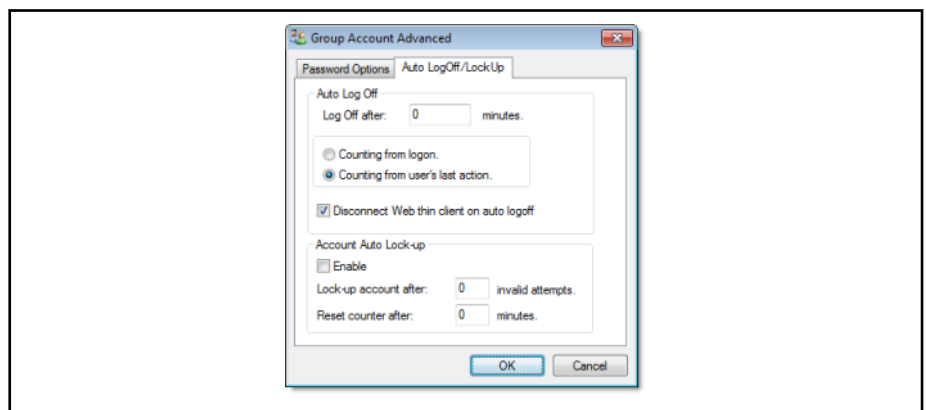


Fig. 12-26: Advanced dialog – Auto LogOff/LockUp

Project Security

Area / Element		Description
Auto Log Off	Log Off after	Number of minutes after which the current user has to be logged off automatically. If this field is left in blank (or with the value 0), the current user is never logged off automatically.
	Counting from logon	If this option is selected, the current user is automatically logged off after the period of time configured in the Log Off after field elapsed since the current user logged onto the system.
	Counting from user's last action	If this option is enabled, the current user is automatically logged off after the period of time configured in the Log Off after field elapsed since the last action (mouse, touchscreen or keyboard action) was performed by the current user.
	Disconnect Web Thin Client on auto logoff	If users logged on via a Web Thin Client are logged off, the client is automatically disconnected from the data server.
Account Lock-up	Enable	Enables the auto lock-up functions described below.
	Lock up account after	Maximum number of times users can try to log on to an account. If a user exceeds the specified maximum number of attempts (provides an invalid password) within the period of time specified in the Reset counter after field, the project automatically blocks the respective user.
	Reset counter after	Defines how long the project waits after an invalid log-on attempt (in minutes) before resetting the log-on attempts counter.



If one user is assigned to more than one group (see [Creating and configuring users on page 374](#)), the group settings may conflict with each other. The correction of the settings depends on their type:

- The settings in the "Group Account" dialog above are "admissible", i.e., the most admissible setting from all user groups applies to the user. For example, if any of the groups can create and modify tags, then the user can create and modify tags.
- The settings in the "Group Account Advanced" dialog (both tabs) are "restrictive", i.e., the most restrictive setting from all user groups applies to the user. For example, if one group has a minimum password size of 8 and another group has a minimum password size of 12, the minimum password size of the user is 12. (For Auto Log Off in particular, **Counting from logon** overrides **Counting from user's last action**.)

12.7 Creating and Configuring Users

To create and maintain accounts for project users, click the **Users** button on the "Security System" dialog.

The "User Account" dialog is displayed.

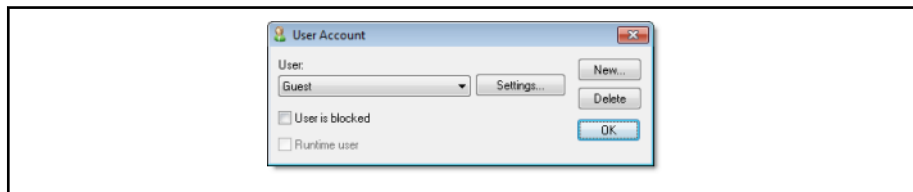


Fig. 12-27: User Account dialog

If no users log on after project initialization (or if the current user logs off), the project automatically logs on the default user (**Guest**). In addition to the de-

fault **Guest** user, there is a **Guest** group, which has default privileges that enable all tasks. It is recommended to evaluate and edit the **Guest** group access rights to specify minimum rights for the start up procedure.

To create a new user, click **New** to open the "New User Account" dialog.

To delete a user, click the **User** combo box button, select the user name from the list and click **Delete**.

To configure a user, proceed as follows:

1. Click on the **User** combo box button and select a user from the list.
2. If necessary, click on the **User is blocked** checkbox to unblock the selected user.



The **Runtime user** option indicates that a user was created during runtime using the [CreateUser on page 536](#) function.

This option cannot be selected for users created using the procedure described here.

3. Click on the **Settings** button to open the "User Settings" dialog:

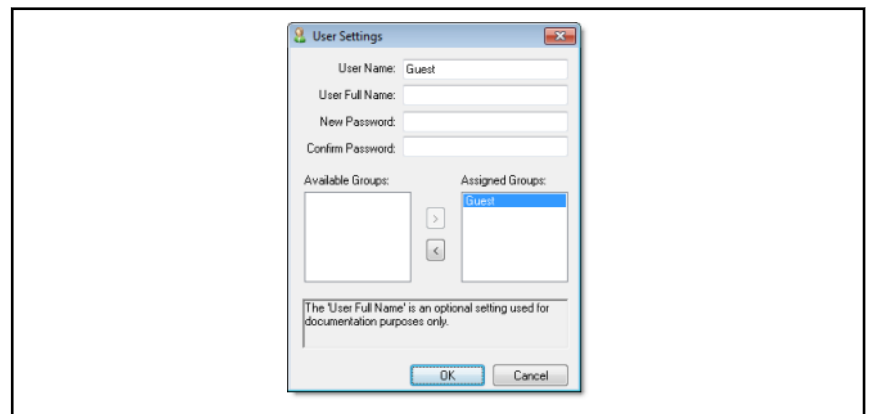


Fig.12-28: User Settings dialog

4. To configure the parameters on this dialog, proceed as follows:
 - **User Full Name** text field ("optional"): Enter the full user name.
 - **New Password** text field: Enter the user password.
 - **Confirm Password** text field: Re-enter the user password.
5. In the **Available Groups** list, select the group(s) to which the user is to be assigned and click > to move these group(s) to the **Assigned Groups** list.
6. To finish, click on **OK** to apply the changes and close the "Settings" dialog.

12.8 Security System Dialog

The main "Security System" dialog is used to manage the project security system after it has been initially configured.

Accessing the dialog

Assuming the project security system has already been enabled (i.e., the Security System Configuration Wizard was used at least once), this dialog can be accessed as follows:

In the WinStudio context menu, select **Tools ► Safety settings**.

Project Security

If one of these actions is carried out "before" the security system has been enabled, the [Security System Configuration Wizard on page 359](#) opens automatically.

If the security system has already been configured and a main password has been set, the password is requested.

The dialog in detail

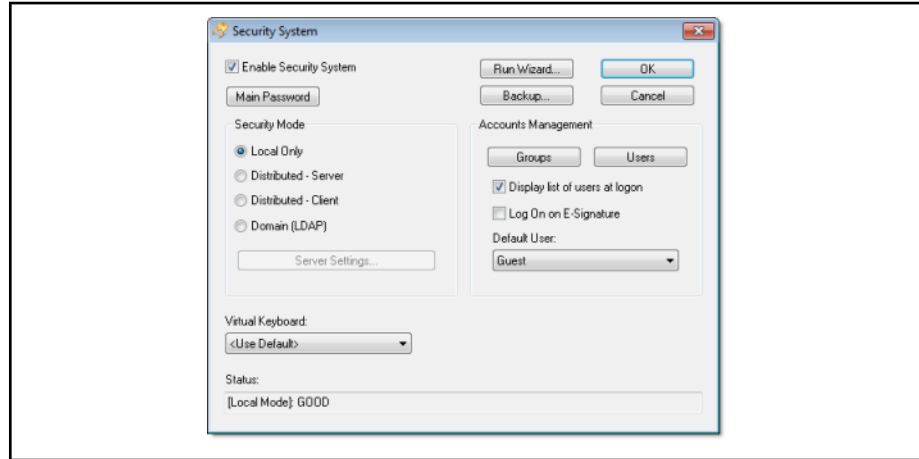


Fig.12-29: Security System dialog

Area / Element		Description
Enable Security System		Indicates whether the project security system is currently enabled. If it is, the specific access rights of users and groups are enforced.
Main Password		Opens a dialog to specify a main administrative password for the entire project.
Security Mode	Mode	The current Security Mode on page 357 of the project.
	Server Settings	Opens the Server Settings dialog on page 362 to configure the server settings for Distributed – Client or Domain (LDAP) .
Run Wizard		Opens the Security System Configuration Wizard on page 359 .
Backup		Opens the Import/Export dialog on page 377 to export or import the security system configuration.
Accounts Management	Groups	Opens the Group Account dialog on page 370 to create and configure groups.
	Users	Opens the User Account dialog on page 374 to create and configure users.
	Display list of users at logon	Displays a list of available users (in the Log On dialog on page 378) if a user is prompted to log on. Users can select from this list instead of entering the user name.
	Log On on E-Signature	Forces users to log on with their own user account when prompted to e-sign an event. If this option is not enabled, the current user account remains logged on regardless of who e-signs the event.
	Default User	This user is automatically logged on when no other user is logged on, e.g., if the previous user times out or logs off manually. Note: The rights of this user are to be heavily restricted, to prevent the project from being left vulnerable.
Virtual Keyboard		The type of virtual keyboard displayed on the client if the user is prompted to log on.

12.9 Backing up the Security System Configuration

The security system configuration of the project can be backed up by exporting it to a file. Configurations can also be imported from a file or from another runtime project.

Exporting the configuration to a file

To export the security system configuration, proceed as follows:

1. In the main [Security System dialog on page 375](#), click on **Backup**. The "Import/Export" dialog is displayed.
2. Click on **Export to file**. A standard "Save As" dialog is displayed.
3. Specify a file name and location for the file and click on **OK**.

The exported file is encrypted with the main password configured in the "Security System" dialog.



The configuration can also be exported during runtime by means of the [ExportSecuritySystem on page 537](#) function.

Importing the configuration from a file

If the [Security Mode on page 357](#) is set to **Local Only**, a configuration can be imported from a previously exported file.

To import the security system configuration, proceed as follows:

1. In the main [Security System dialog on page 375](#), click on **Backup**. The "Import/Export" dialog is displayed.
2. Click **Import from file**. A standard "Open" dialog is displayed.
3. Locate the configuration file (*.dat) to be imported and click on **OK**. The main password is requested for configuration.
4. Enter the password and click on **OK**. The "Import from File" dialog is displayed.
5. Select an import method:
 - **Import only settings that do not conflict:** Merge the imported settings with the current project settings. In the event of a conflict, the current settings are kept.
 - **Import all settings and replace conflicts:** Merge the imported settings with the project settings. In the case of conflicts, use the imported settings.
 - **Replace the current settings:** Completely replace the current project settings with the settings imported from the file.
6. Click on **OK**.



The configuration can also be imported during runtime by means of the [ImportSecuritySystem on page 542](#) function.

Importing the configuration from another project

If the [Security Mode on page 357](#) of the project is set to **Distributed – Server**, a configuration can be imported from a different project if:

- The security mode of the other project is set to **Distributed – Client** and its server settings are configured to use the project as server; and
- The other project is currently running on the same network.

To import the security system configuration, proceed as follows:

1. In the main [Security System dialog on page 375](#), click on **Backup**. The "Import/Export" dialog is displayed.
2. Click on **Import from client station**. The **Import Security from Client Station** dialog is displayed. The dialog shows a list of runtime projects that

Project Security

are using the project as security system server. Each project/client list includes a time stamp that shows when it last cached the security system configuration.

3. Select a client station and click on **Import from client**. The main password is requested for configuration.
4. Enter the password and click on **OK**.

12.10 Logging On/Off

If the project security system has been enabled and the default "Guest" user rights have been restricted, logging on is required to fully use the development application and/or the runtime project.



The project security system has to be enabled before this function is available.

To log on to the development application, in the WinStudio context menu, select **Tools ► Log On / Log Off Users**.

To prompt a user to log on to the runtime project, proceed as follows:

- Open the [LogOn on page 556](#) function at a position where an expression can be configured – for example, draw a [button object on page 176](#) in a screen and apply the [command animation on page 220](#) to it, so that pressing the button shows a logon prompt; or
- Select the **Log On on E-Signature** option (in the main [Security System dialog on page 375](#)) to force the user to log on when performing an action that requires an e-signature.

In either the development application or the runtime project, the Log On dialog is displayed:

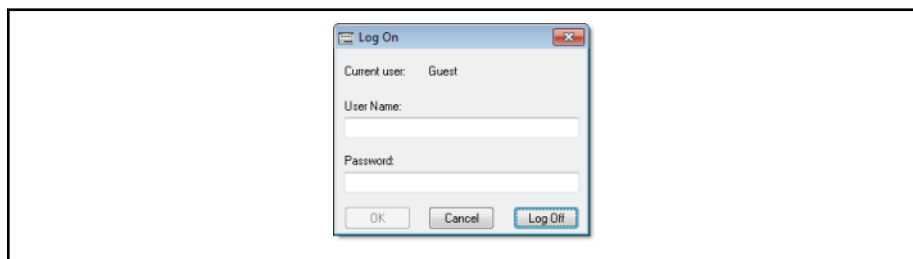


Fig. 12-30: Log On dialog

Use this dialog as follows:

- To log on with an own account, enter the user name and password in the respective fields and click on **OK**.
- To log on as the default "Guest" user, enter "Guest" in the **User Name** field and click **OK**.



By default, "Guest" has no password and the **Password** field can be left empty. However, if the password was changed or the security settings are received from a server (either Distributed or Domain), the "Guest" password has to be reentered.

- To log off, simply click on **Log Off**.
The default user (typically "Guest", however, this may be changed in the main [Security System dialog on page 375](#)) is automatically logged on to replace the user.

NOTICE

If the [Security Mode on page 357](#) is set to Domain (LDAP) and a user created on the LDAP server is required to change the password when logging on to the domain for the first time, this has to be done before the user is able to log on to the "WinStudio" project.

12.11 Blocking or Unblocking Users

Individual users in the project security system may be completely blocked from accessing the project and respectively, blocked users can be unblocked.

To block users, proceed as follows:

- By manually selecting the **User is blocked** option in the [User Account dialog on page 374](#);
- By using the [BlockUser on page 535](#) function during runtime; or
- Automatically, if the respective user enters the wrong password too many times. (The number of attempts allowed is configured in the [Group Account dialog on page 370](#).)

To check whether a user is blocked, use the [GetUserState on page 541](#) function.

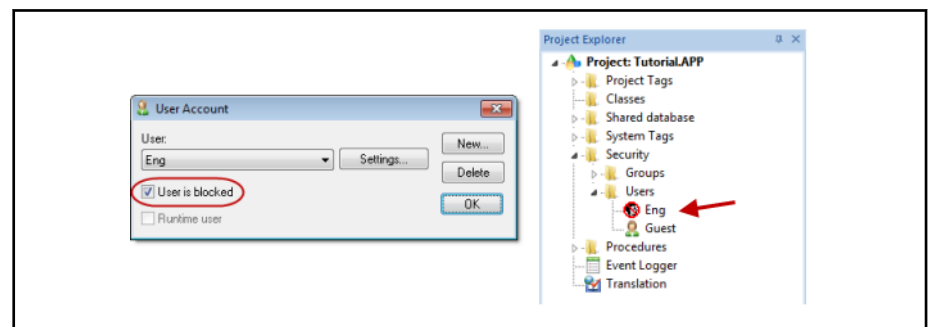


Fig. 12-31: Unblock users

To unblock a blocked user, proceed as follows:

- Disable the **User is blocked** option in the "User Account dialog"; or
- Use the [UnblockUser on page 545](#) function during runtime.

12.12 Password Protection for Screens, Symbols and Worksheets

In the Project Explorer, screens, symbols, and worksheets can be password-protected. Individual passwords can be assigned to each file or a single password to all files in the project.

Almost all project files are automatically encoded to prevent unauthorized analysis by third-party tools. (Screen files are not encoded as decoding them during runtime would decrease performance.)

However, files can be additionally password protected to prevent unauthorized changes or re-use by other "WinStudio" project developers.



These passwords are always case sensitive.

Assigning a password to a single file

To assign a password to a single project file, proceed as follows:

Project Security

1. In the Project Explorer, right-click the respective file. In the context menu, click on **Password Protection**. The "Edit Password Protection" dialog is displayed.
2. Specify the new password and reenter it for confirmation.
3. Click on **OK** to close the dialog.

The file is now protected. The next time the file is opened, the password is requested.

Clearing the password from a single file

To clear a password from a single project file, proceed as follows:

1. In the Project Explorer, double-click on the respective file to open it. The password is requested.
2. With the file open for editing, right-click on the file in the Project Explorer and select **Password Protection** in the context menu. The "Edit Password Protection" dialog is displayed.
3. Leave the **New password** and **Confirm password** boxes empty.
4. Click on **OK** to close the dialog.

The file is no longer protected. The file can be opened without entering the password.

Assigning a password to all files

To assign a single password to all project files, proceed as follows:

1. In the WinStudio context menu, select **Verify Application**. The "Verify Application" dialog is displayed.
2. Click on **Set password for all files**. The "Edit Password Protection" dialog is displayed.
3. If applicable, enter the current password for the project.
4. Specify the new password and reenter it for confirmation.
5. Click on **OK**. The verification process is continued.
6. Click **Close** to close the "Verify Project" dialog.

Now, all project files are protected. If a file is opened, the password is requested.

Clearing the password from all files

To assign a single password to all project files, proceed as follows:

1. In the WinStudio context menu, select **Verify Application**. The "Verify Application" dialog is displayed.
2. Click on **Set password for all files**. The "Edit Password Protection" dialog is displayed.
3. Enter the current password for the project.
4. Leave the **New password** and **Confirm password** boxes empty.
5. Click on **OK**. The verification process is continued.
6. Click on **OK** to close the "Verify Application" dialog.

The project files are no longer protected.

13 Translation of User Texts

13.1 Translation of User Texts, General Information

All texts used in WinStudio screens, screen objects, alarm messages, etc., can be translated and switched between different languages during runtime. The following steps are required to enable switching between languages in projects:

- Adding project languages to the IndraWorks project
- Translating the application texts to the added languages
- Activating a language at runtime

The individual steps are described in the following chapters.

13.2 Adding a Project Language

The Translation Table is used to manage the languages to translate the project to. The list of available languages corresponds to the languages available in the IndraWorks project.

To add a language, proceed as follows:

1. In the IndraWorks main menu, select **Project ► Language ► Manage Project Languages**

A dialog is displayed to add the required languages by clicking on "Add".

2. Open the "localization" worksheet in the "WinStudio\Tasks" folder of the IndraWorks Project Explorer. In the "localization" worksheet, all character strings that can be translated are displayed in the Source column of the table.

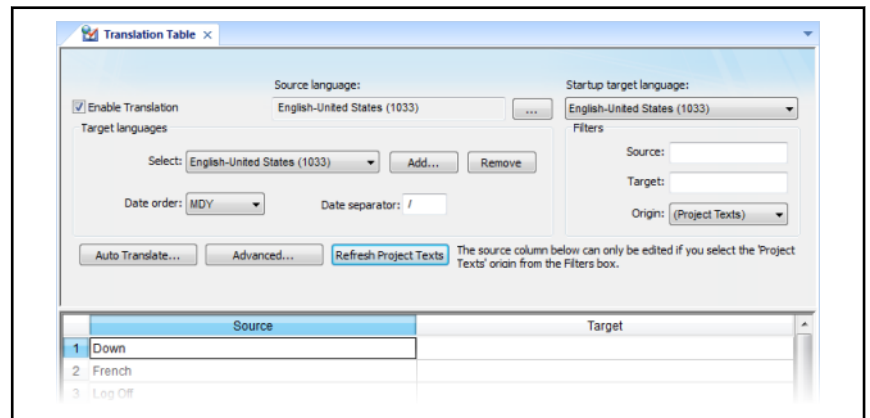


Fig. 13-1: Translation Table worksheet

3. In the "Target languages" pane, select the languages to which the application texts are to be translated. The list of available languages corresponds to the IndraWorks project languages.
4. Ensure that the **Enable Translation** option is selected.
5. Configure the **Date order** and **Date separator** as required for the target language. For example, the **Date order** for "English-United States" (i.e. American English) is typically "MDY" and the **Date separator** is typically "/", resulting in a date format of "MM/DD/YYYY".

13.3 Translating Application Texts to Additional Languages

After adding the required project languages to the project, these languages can be used in the WinStudio application to translate the application texts respectively.

To translate application texts, proceed as follows:

1. Open the "localization" worksheet in the "WinStudio\Tasks" folder. In the "localization" worksheet, all character strings that can be translated are displayed in the Source column of the table.
2. In the "Target languages" pane, select the languages to which the application texts are to be translated. The list of available languages corresponds to the IndraWorks project languages.
3. Enter the translation of the original texts for the respective target language into the "Target" column. In order to only edit texts from a certain context of the WinStudio application, select the respective type, e.g. Alarm Control, in the "Origin" combo box.
4. Use the **Filters** fields to search the worksheet for a specific character string. When entering a few characters, the list is filtered dynamically and only displays suitable strings.
5. Save and close the worksheet.

Changes made to the Translation Table do not apply before the [Setting the Project Language at runtime on page 383](#) function is executed or the runtime project is restarted.



The Translation Table is saved as a tab-delimited text file in the project folder, at

[...]\project_name\Web\Translation.trn.

If required, this file can be opened and directly edited with Microsoft Excel.

13.4 Activating a Language at Runtime

13.4.1 Activating a Language at Runtime, General Information

When activating a project language during runtime, the application type has to be generally considered:

- **HMI or MTX application**

If an HMI or MTX application is projected, i.e., the application is executed within IndraWorks Operation, no additional configuration is required to activate a language. The application is already set for language selection and the active language can be either selected in IndraWorks Operation via the main menu under **Tools ► Options** or by pressing an F/M button.

The selected language applies for all WinStudio screens.

- **WinStudio application**

For WinStudio applications, the following steps are required to activate a language at runtime:

1. Select a starting language to be activated if the application is started.
2. Add control elements at suitable locations in the application to change the active language during runtime.

The two steps are described in the following chapters.

13.4.2 Setting the Project Language at Program Start

If multiple languages are configured for the project, the project starting language at runtime has to be specified.

Before, at least one target language has to be added to the Translation Table.

For further information, refer to [Adding a Language on page 381](#).

1. Open the Translation Table by double-clicking on the "Localization" element in the "WinStudio\Tasks" folder.

The Translation Table worksheet is opened for editing.

2. Select the project starting language in the **Startup target language** list. The list of available languages includes the source language in which the project has been developed as well as all added languages.
3. Save and close the worksheet.

13.4.3 Setting the Project Language at runtime

To activate a project language, WinStudio provides the command [SetLanguage on page 607](#). By means of this function, the project language can be set at runtime where expressions can be configured.

To use this function, at least one target language has to be added to the Translation Table. For further information, refer to [Adding a Language on page 381](#).

The "SetLanguage" function requires a numerical code for the target language parameter. The codes are shown in parentheses in the list of languages – for example, "English-United States" is "1033".

The following very basic examples illustrate how to draw two button objects that switch the project language between English and French.

1. In the "WinStudio" folder, click on a WinStudio screen to open it for editing.
2. In the "WinStudio Screen Tools" tool window, click on **Button**.
3. Draw a button object into the WinStudio screen.
4. Double-click on the button object. The "Object Properties" dialog is displayed.
5. In the **Caption** box, type "English".
6. Click on **Command**. The command animation properties are displayed in the dialog.
7. In the first row of the **On Down** tab, in the **Expression** field, enter Set - Language(1033).

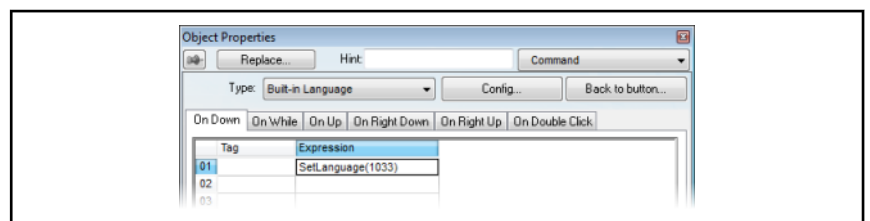


Fig. 13-2: Setting the project language (1)

8. Close the "Object Properties" dialog
9. Duplicate the Button object, either by copy-and-paste or by Ctrl+Click.

Translation of User Texts

10. Repeat steps 4 through 8, replacing the caption with "French" and the expression with `SetLanguage(1036)`.

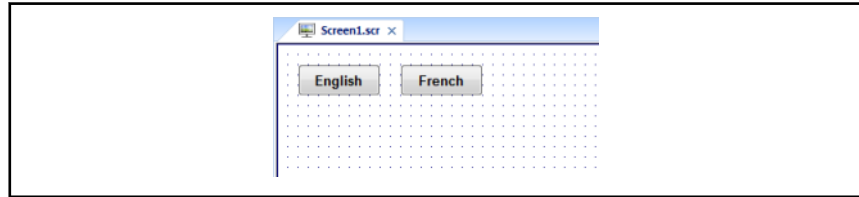


Fig. 13-3: Setting the project language (2)

11. Save and close the WinStudio screen.

During project runtime, clicking each button sets the language of the entire project, using the translated text from the Translation Table.

13.5 Disabling Translation of Selected Screen Objects

By default, translation is enabled for all screen objects that have text to be translated. However, translation of selected objects can be disabled if the original text has to be kept.

1. Double-click on a screen object to open its "Object Properties" dialog.
2. In the dialog, look for the **Enable Translation** option. If the option is not available in the basic properties of the object, click on **Advanced** to open the advanced settings.
3. Disable the **Enable Translation** option.
4. Close the "Object Properties" dialog

Once the option is disabled on an object, its text is no longer translated during project runtime.

The text is still added to the Translation Table and can be processed though the automatic translation engine along with all other project texts, however, the resulting translation is not applied to the object during runtime.

14 Testing and Debugging Projects

14.1 Output (LogWin)

Use the "WinStudio Log" window to display error analysis messages generated during project runtime. The window displays OPC, DDE, and TCP/IP transactions, module activations, trace messages, and so on.

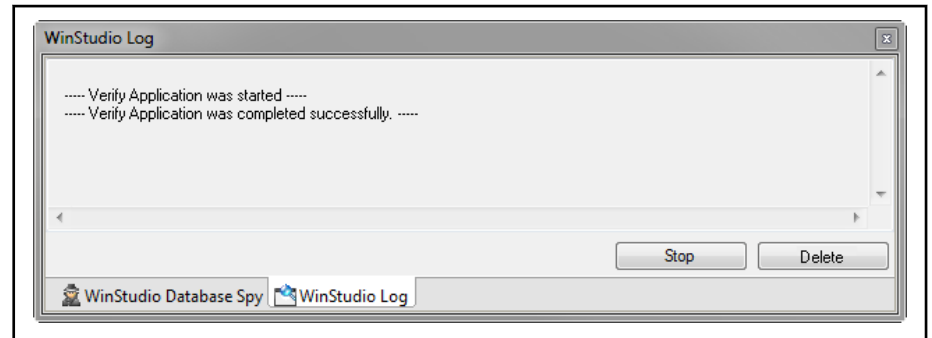


Fig. 14-1: Example of output window

The "WinStudio Log" window is dockable, i.e. it can be moved to another location in the development environment. To do so, click on the title bar and drag the window to a new location. Release the mouse button to attach or "dock" the window to its new location.

Configuring the log to show additional information

By default, the log shows only error analysis and error messages — that is, messages indicating that the project is not running properly. If the log displays **all** messages created by "WinStudio", the output quickly becomes confusing.

To configure the messages displayed in the log, proceed as follows:

1. Right-click anywhere in the "WinStudio Log" window and then click **Settings** in the context menu. The "Log Settings" dialog is displayed:
2. On the "Log Options" tab of the dialog, select the specific types of messages that are to be displayed in the log.

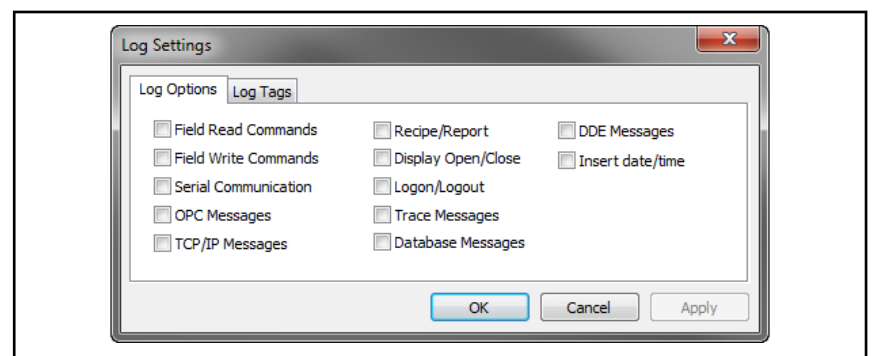


Fig. 14-2: Log Settings - Options tab

Options	Description
Field Read Commands	Displays all field read commands that are sent to connected devices
Field Write Commands	Displays all field write commands that are sent to connected devices.
Serial Communication	Displays messages generated by configured device drivers on page 333 .
OPC Messages	Displays messages generated by OPC communications on page 342 .

Testing and Debugging Projects

Options	Description
TCP/IP Messages	Displays messages generated by TCP/IP communications on page 351 .
Recipe/Report	Displays messages of the tasks for recipe on page 319 and reports on page 320 .
Display Open/Close	Displays detailed information whenever a screen on page 156 is opened or closed: Disk Load Time: Time to load the screen file from the disk into memory Open Time: Time to load the WinStudio screen, including initialization tags used in the screen and running any "OnOpen" scripts or functions Total Load Time: Total time to load the screen (includes disk load time and open time above). "First Draw" Time: Time for the first drawing of screen objects "First OnWhile" Time: Time for running any "OnWhile" scripts or functions for the first time Total Open Time: Total time to open the screen (includes First Draw Time and First OnWhile Time as described above). Close Time: Time to close the screen, including finalizing tags used in the screen and running any "OnClose" scripts or functions Total Close Time: Total time to close the screen, including the time to write the screen file on the disk This information can be used to analyze runtime performance on low-end end systems. If a particular step of opening or closing takes an unusually long time, it can be identified and redesigned
Logon/Logout	Displays a message whenever a user logs on or logs out (For further information, refer to Security on page 357 .)
Trace Messages	Shows messages generated by the Trace() function. This function is used to generate customized messages from the project to trace execution of the application
Database Messages	Displays messages generated by ODBC on page 321 and ADO.NET on page 328 database interfaces
DDE Messages	Displays messages generated by DDE communications on page 352 .
Insert date/time	Add a timestamp to each message

Tab. 14-1: Log Settings - Options tab

3. In the "Log Tags" tab of the dialog, click **Add** to browse for [project tags on page 106](#).

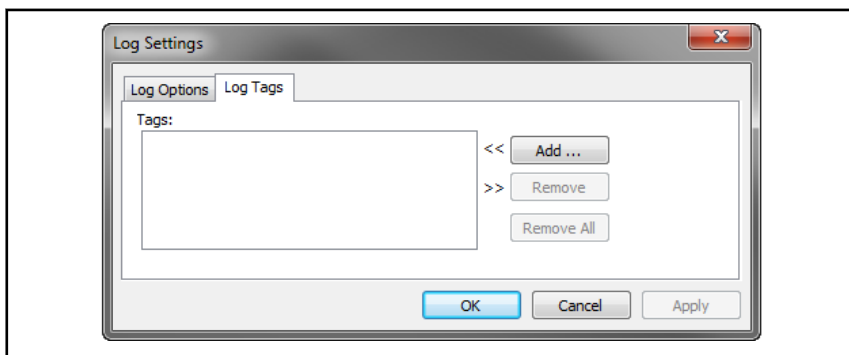


Fig. 14-3: Log Settings — Tags tab

The Log pane of the LogWin window displays a message if the value of a selected tag changes.

4. Click **OK** to save the settings and close the "Log Settings" dialog.

14.2 Database Spy

The "Database Spy" window is a tool for error analysis to display and change tag values and execute and test functions and mathematical expressions.

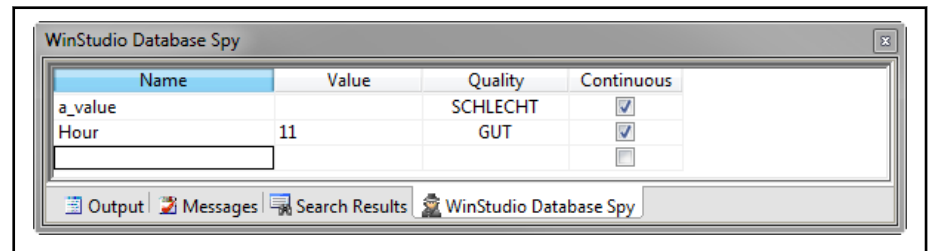


Fig. 14-4: Example of Database Spy window

The window displays the following information:

- **Name:** Indicate an application variable, an internal variable or an expression to be monitored
- **Value:** Displays the value returned by the tag/expression
- **Quality:** Displays the quality (GOOD or BAD) of the value returned by the tag/expression
- **Continuous:** Select this option to have the project continuously evaluate the tag/expression

In order to monitor a variable, enter its name into the column or click on an empty field in the "Name" column and open the context menu. By means of the "Insert variable or expression" function, the browser dialog can be opened alternatively and a variable can be selected.



The Database Spy is dockable, i.e. it can be moved to another location in the development environment.

Click on the title bar and drag the window to a new location.

Release the mouse button to attach or dock the window to its new location.

14.3 Using the LogWin Module

This module provides a continuous record of activities and tags for debugging over long periods of time. It creates a file in which the results of the data collection can be stored. The size of this file increases until the logging process (data collection) is stopped.

Use the LogWin module (local and remote) to record [DDE on page 352](#), [OPC on page 342](#) and [TCP/IP on page 351](#) transactions, activate modules, trace tags, etc.

To enable the LogWin module as a part of the project, proceed as follows:

1. In the context menu of the "WinStudio" folder, select **Tools ► Status**. The "Execution Tasks" dialog is displayed:

Testing and Debugging Projects

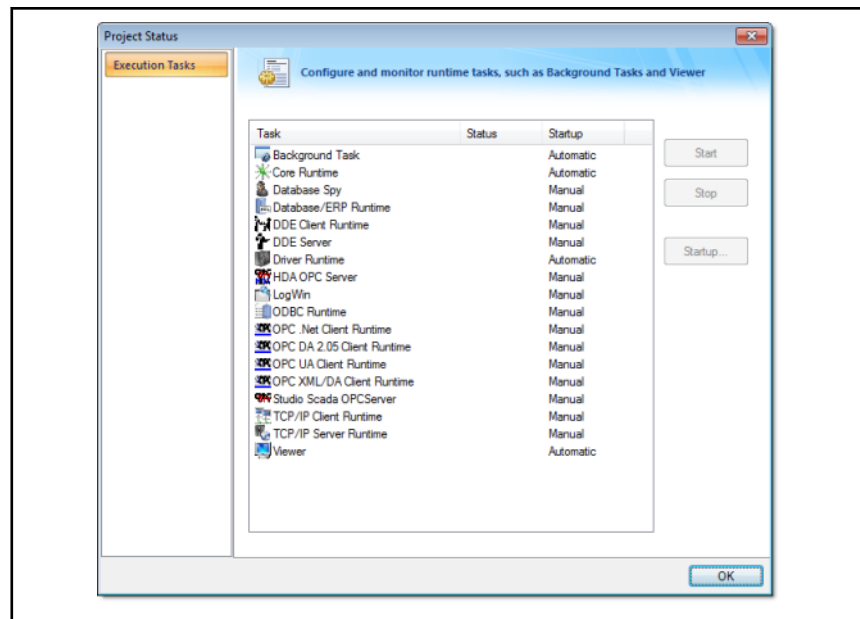


Fig. 14-5: Execution Tasks dialog

2. Select LogWin from the **Task** list.
3. Select the LogWin task and click on **Start**. The "Start" dialog is displayed.

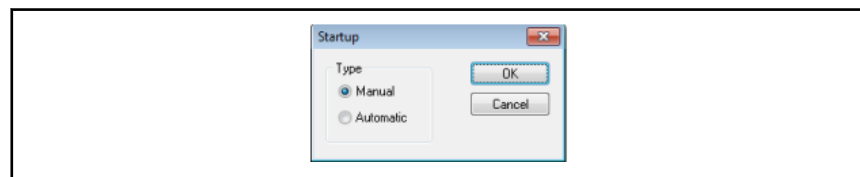


Fig. 14-6: Startup dialog

4. Enable the **Automatic** radio button.
 5. Click on **OK** to apply the selection and close the "Startup" dialog.
 6. Click on **OK** to close the "Execution Tasks" dialog.
- To start the LogWin module directly on a Windows Embedded device, in the CEView menu bar, select **Tools ► LogWin**.
 - To activate logging on a [Thin Client on page 391](#) select the **Log** function in the "Web" Project Settings. The Web Settings dialog can be found in the WinStudio context menu under **Tools ► Web Settings**.

14.4 Using Remote Tools

14.4.1 Remote Database Spy

The **Remote Database Spy** tool can be used to monitor the [Database Spy on page 84](#) of a "WinStudio" project executed on a remote computer. The "Database Spy" execution task has to be enabled for the project and the remote computer has to be at [runtime on page 60](#).

To use the "Remote Database Spy" tool, proceed as follows:

1. In the WinStudio context menu, click on **Tools ► Remote Database Spy**. The "Remote Computer" dialog is displayed.
2. Enter the IP address of the remote computer, as illustrated below.

Testing and Debugging Projects

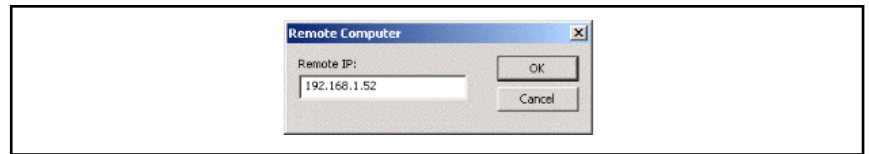


Fig. 14-7: Entering a remote IP



The IP address **192.168.1.52** is only an example. Verify the IP address of the computer to which the connection is to be established.

- Click on **OK** to connect to the specified address. If the connection is successful, the "Remote Database Spy" window is displayed.

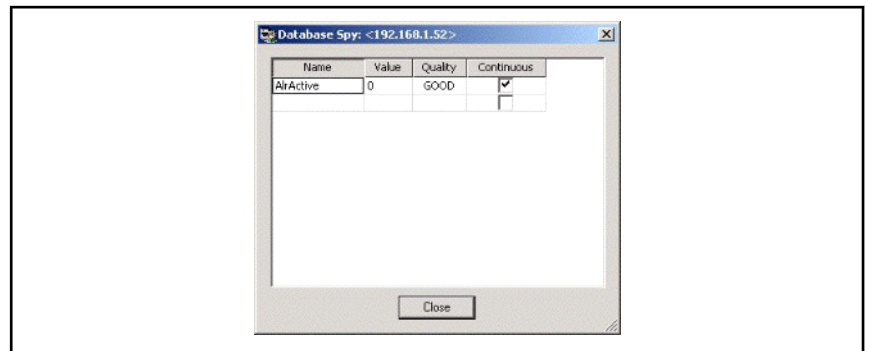


Fig. 14-8: Remote Database Spy



Tags cannot be added or removed remotely. The tag list of the Database Spy has to be configured on the remote computer itself.

To finish, click on **Close** to disconnect from the remote computer.

14.4.2 Remote LogWin

The **Remote LogWin** tool can be used to monitor the [Output log \(LogWin\)](#), [page 85](#) of a "WinStudio" project executed on a remote computer. The "LogWin" execution task has to be enabled for the project and the remote computer has to be at [runtime](#), [page 60](#).

Using the Remote LogWin tool:

- In the WinStudio context menu, click on **Tools ► Remote LogWin**. The "Remote Computer" dialog is displayed.
- Enter the IP address of the remote computer, as illustrated below.

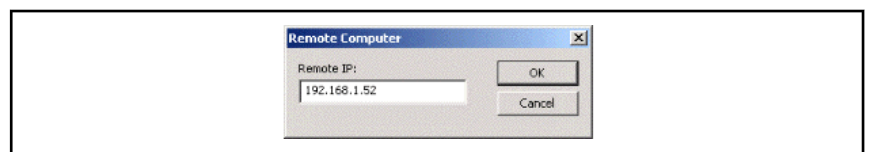


Fig. 14-9: Entering a remote IP



The IP address **192.168.1.52** is only an example. Verify the IP address of the computer to which the connection is to be established.

- Click on **OK** to connect to the specified address. If the connection is successful, the "Remote LogWin" window is displayed.

Testing and Debugging Projects

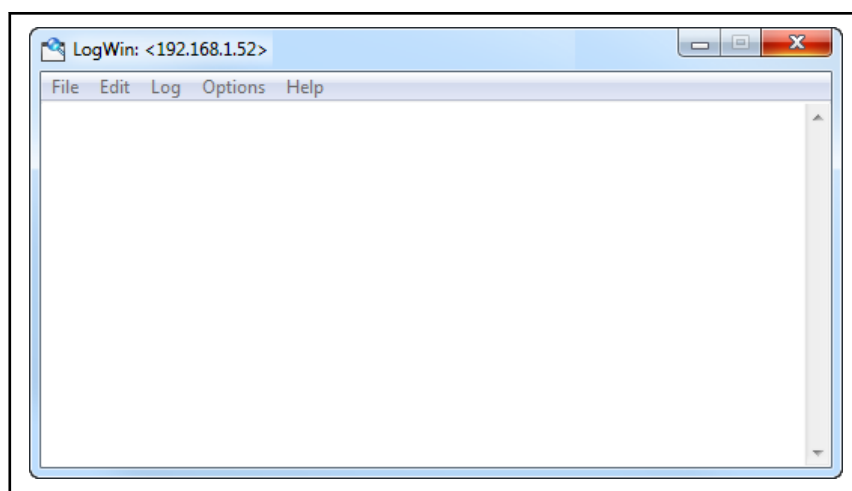


Fig. 14-10: Remote LogWin

After finishing, just close the window to disconnect.

15 Provision as a Web Application

15.1 Introduction to Thin Clients

"WinStudio" is based on a client/server architecture and supports both Thin Clients and Thick Clients. The selected type of client architecture depends on the system requirements.

Thick Client A Thick (Fat) Client is a computer that performs most, if not all, processing activities.

Thick Clients have sufficient processing power, memory, graphics, etc. to run the project. Thick Clients can exchange data with the server as required (e.g., data logging, program synchronization). Servers can also be Thick Clients for other servers.

Thin Client

Thin (Lean) Clients are computers that depend primarily on a central server for processing activities. Thin Clients require a processor with minimal memory, a browser (Internet Explorer) and a network connection (e.g., to the Internet or Intranet). The term "Web Client" refers to a Thin Client solution. "WinStudio" enables selecting the type of browser that suits best the runtime environment.

- Microsoft Internet Explorer. This type of browser is recommended for desktop or remote access environments.
- "Bosch Rexroth" Secure Viewer, a dedicated viewer for "WinStudio" projects. The Secure Viewer cannot be used to browse other websites and can only connect to a defined "WinStudio" server. The Secure Viewer is intended for plant floor environments where browsing other websites is not required or desired.

This chapter concerns the application of Thin Clients with "WinStudio". Thick Clients are covered in a subsequent chapter.

Thin Clients in "WinStudio"

"WinStudio" enables creating screens that can be visualized on remote stations with a regular web browser (e.g., Internet Explorer). The station where the graphical interface (screens) can be visualized on the web browser is called Thin Client. The Thin Client (browser) is the host.

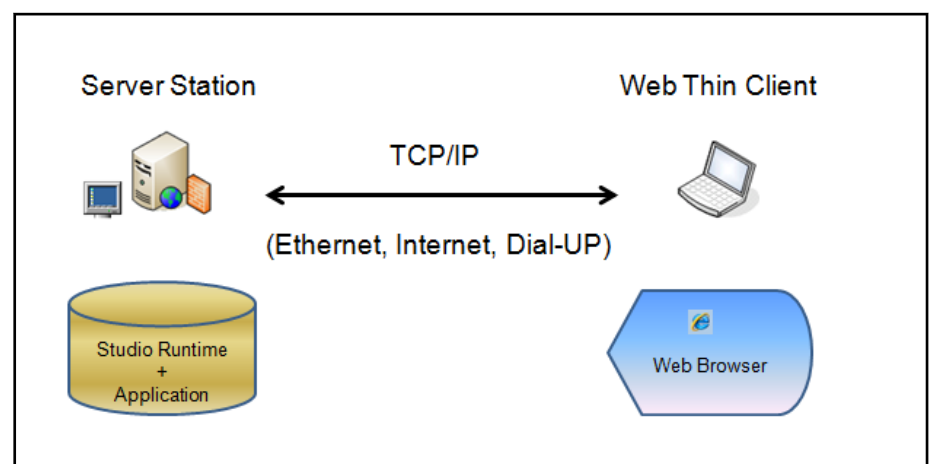


Fig. 15-1: Typical Thin Client architecture

"WinStudio" is only installed on the server station. Additionally, the project (screen files, tags database, configuration worksheets, etc.) is only stored on the server. The development application or the runtime project do not need to be installed on the Thin Client. This solution provides a high level of flexibility

Provision as a Web Application

as the screen and online/history data of the server can be accessed from any computer physically linked to the server station (via a TCP/IP link) without installing the development application or the runtime project on the Thin Client.

All background tasks (Math, Scheduler, etc.) and communication tasks (e.g., Driver, OPC, DDE TCP/IP) are executed only on the server station. Thin Clients can load graphical interfaces configured on the server (screens with objects and animations) and display the online values of the tags configured on the server as well as history data (Alarm, Event and Trend history data).



Since VBScript and the "WinStudio" scripting language can be associated with a screen as well as a command animation for an object, these scripts can be executed on Thin Clients.

Any computer or device (e.g., a PDA with Windows Mobile 5.0) running Microsoft Internet Explorer 6.0 (or later) or the Bosch Rexroth Secure Viewer program can be used as Thin Clients for projects. Additionally, "WinStudio" provides a sophisticated **Security system** to prohibit unauthorized access to the project.

From Thin Clients, it is possible to visualize data from the server and to change set points, acknowledging alarms and/or sending commands to the server. When configuring the project, all commands can be optionally enabled/disabled from the Thin Client to the server. Even if the server has disabled all commands from the Thin Client, the Thin Client can still read data from the server, however, not send any data to the server.

Thin Client – Competitive advantages

"WinStudio" is built on a client/server architecture that supports **true** Thin Clients. This capability is integrated into "WinStudio" and is not an add-on. This means that:

- The "WinStudio" server supports a large number of concurrent Thin Clients (up to 128 IE-based and 128 Secure Viewer-based). Each Thin Client can access the same or different screens as other Thin Clients.
- The "WinStudio" server knows which display is accessed by each Thin Client and automatically "pushes" all updated tag values to the Thin Client. This way, the display of the Thin Client is continuously updated and the screen does not need to be updated manually.
- The "WinStudio" server can support runtime language switching for each Thin Client. This means that one Thin Client can display a screen in English while a different Thin Client can display the same screen in Spanish.

Many competitive products offer either a static display on a Thin Client (i.e., it has to be "refreshed" manually to get the latest data), a Terminal Server solution (requires the server to build multiple instances of the project to support each Thin Client) or a Thin Client solution similar to "WinStudio" but with expensive "add-on" software products.

Other Thin Client advantages include:

- Optional Secure Viewer that does not enable navigation outside the "WinStudio" project. Secure Viewer is not based on Microsoft Internet Explorer.
- Option of running VBScript and the built-in scripting language on the Thin Client
- Option of building Thin Client projects using Windows Embedded devices as server.
- Ability to support redundant web servers and data servers with automatic switchover.

Provision as a Web Application

Thin Client Licensing The maximum number of Thin Client stations connected simultaneously to the server depends on the settings of the license installed on the server. No licenses need to be installed on the Thin Client. For further information, refer to [License settings on page 43](#).

15.2 Creating a Simple Thin Client Program

As described in the following, there are different ways to build Thin Client programs. The following procedure enables developing a simple, unsecured Thin Client program.

Procedure A: Thin Client using NTWebServer, Local loopback

An easy way of developing and testing a Thin Client program is using the light-weight web server **NTWebServer** on the development PC, initiating the runtime project and using Microsoft Internet Explorer on the same PC to display the web pages.

Step 1: Configure the "WinStudio" settings

1. Open the configuration dialog for Web settings by selecting **Tools ► Web Settings** in the WinStudio context menu.

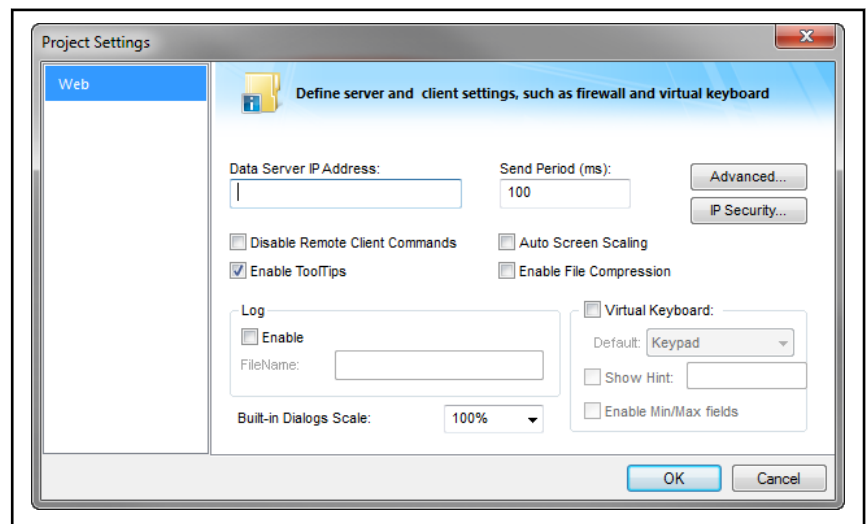


Fig. 15-2: WinStudio settings, Thin Client

- In the Data Server IP Address field, enter **127.0.0.1**
- Make sure that the **Auto Screen Scaling**, **Enable File Compression** and **Enable Tool Tips** checkboxes are enabled.
- Click on the **IP Security Button**. Make sure that the **Enable** checkbox is disabled. Click on **OK**.
- Click on **Advanced**. Make sure that the **Web Tunneling Gateway Enabled** checkbox is disabled. Click on **OK**.

Provision as a Web Application

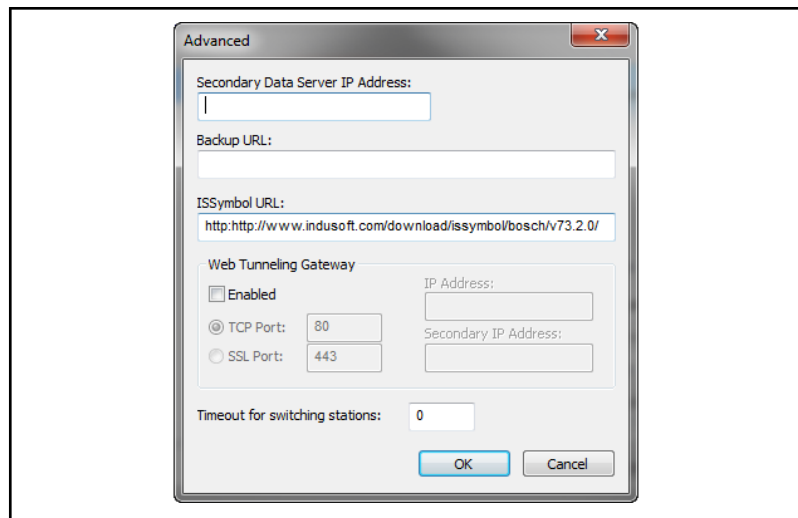


Fig. 15-3: WinStudio settings, Thin Client, Advanced

2. Open the task configuration (via **Tools ► Status** in the WinStudio context menu) and make sure that the **TCP/IP server** task is set to Automatic.

Step 2: Configure the database

1. In the "Variables" folder, open the **Application Tags** editor.
2. Make sure that the application tags are properly set to either server or local. If the tags are to be provided to the web client, set them to **Server**, otherwise to **Local**.

Step 3: Develop screens and create HTML screens

1. Develop a WinStudio screen. Depending on the Thin Client screen size, it may be necessary to develop a separate set of screens with a different resolution. Auto Screen Scaling is supported (enabled in a prior step), however, has natural limitations.
2. Save and close all project screens.
3. Make sure that a startup screen has been defined (on the **Viewer** tab of the WinStudio Properties dialog).
4. Individual screens can be saved as HTML by selecting the **Save as HTML** or **Save Screen Group as HTML** option in the application menu or saving all screens and screen groups as HTML by selecting **Save All as HTML**.
5. If only one screen is updated and, particularly, if the configuration of the web settings is changed, the function "Verify Application" is to be applied (in the "WinStudio" folder context menu).
6. The HTML pages (web pages) are stored in the "Web" subfolder of the project folder.

Step 4: Install the NTWebServer

1. NTWebServer is the "Bosch Rexroth" lightweight web server for Windows NT/XP/2000/Server 2003/Vista environments. **NTWebServer.exe** can be found at [...]WinStudio\ Bin
2. Copy **NTWebServer.exe** from the \Bin folder and paste it into the "Web" subfolder of the project folder.
3. In the "Web" subfolder, double-click on NTWebServer to start it.
4. A new window opens. A message is to be displayed at the bottom edge indicating that NTWebServer is ready-to-receive. If a message is dis-

Provision as a Web Application

played indicating that NTWebServer failed to open a socket, it is most likely caused by Microsoft IIS (Web Server) running in a background mode. If this is the case, IIS has to be stopped and NTWebServer re-started.



Web servers typically work or "listen" on the TCP/IP port 80 of a computer. Only one running process can receive a defined port. Respectively, if other processes, e.g., SCADA third-party software, is already receiving on port 80, this process might collide with the server process. Either, one of the processes has to be configured to receive from a different port or the Task Manager has to be used to end the conflicting process. If the conflicting process cannot be identified, open the Windows command prompt and enter the following command to get a list of all networking processes:

```
netstat -a -o
```

Step 5: Start the runtime project

1. In the WinStudio context menu, click on **Start application**.

Step 6: Launch Microsoft Internet Explorer and connect to the web server

1. Click on the **Start** button (or Alt tab) to access the Microsoft Internet Explorer program.
2. Start Internet Explorer and enter the startup address (home web page), e.g., `http://127.0.0.1/startup.html`.
3. Note that the **startup.sg** (or whatever the startup display or screen group name is) has an HTML file extension when accessed from Internet Explorer.
4. Sign on as **Guest** with no password, provided security has not been enabled.

Procedure B: Thin Client using NTWebServer, network IP

Once Procedure A has been completed, the next step is to enable network connected Thin Clients instead of using the local loopback. The IP address of the PC has to be known, as seen from the network. This information can be obtained from the Network Settings in the Control Panel or by entering the **GetComputerIP()** function in a Rectangle Object on a screen and running the project.

Step 1: Configure the "WinStudio" settings

1. Stop all active runtime projects.
2. Open the Web Configuration dialog and enter the IP address of the computer into the Data Server IP address field (as viewed from the network), e.g., 192.168.1.100.
3. Execute the **Verify Application** tool. This changes the data server IP address on the web pages, so that the Thin Client automatically exchanges data with the correct data server.

Step 2: Start the runtime project

1. In the WinStudio context menu, click on **Start application**.
2. Ensure that NTWebServer is still running.

Step 3: Launch Microsoft Internet Explorer and connect to the web server

1. Click on the **Start** button (or Alt tab) to access the Microsoft Internet Explorer program.

Provision as a Web Application

2. Start Internet Explorer and enter the startup address (home web page), e.g., `http://192.168.1.100/startup.html`.
3. Sign on as **Guest** with no password, provided security has not been enabled.

15.3 Underlying Technology

In "WinStudio" projects, there are several components used to implement the Thin Client capability.

These include:

Data server	The data server is integrated into the "WinStudio" runtime. The data server has direct access to the "WinStudio" Project Tags Database (runtime) and is responsible for working with ISSymbol to make sure any tag data displayed on a web page at any Thin Client is updated with the latest value(s). "WinStudio" can support a backup or secondary data server that is used if the primary data server becomes unavailable. The Thin Client automatically switches over to the secondary server without user intervention required.
Web server	The primary web server is responsible for providing web pages on demand (i.e., when requested by the client) through navigation to various project screens by the Thin Client. The web server communicates with the Thin Client via HTTP protocol over TCP/IP. With SSL (Secure Socket Layer) encoded communication can be enabled. The web server does not need to be on the same PC as the "WinStudio" runtime project. In fact, the web server could be a non-Windows corporate web server. However, the web server requires access to the HTML files that are the project web pages. "WinStudio" supports a secondary web server that is automatically switched to (by the Client) in case the primary web server becomes unavailable.
Web browser	The web browser is located on the Thin Client PC and provides the graphical interface function with the user. Web pages (HTML) are passed to the browser via demand ("pull") and data is delivered ("pushed") to the browser by the data server whenever a tag or tags referenced on the screen displayed on the web client is updated in the tag database.
ISSymbol	ISSymbol is a "Bosch Rexroth"-provided ActiveX V control element that facilitates the interaction between the browser on the web client and the web server as well as the data server. There are ISSymbol ActiveX control versions for Windows XP/Vista/7 and all Windows Embedded platforms. The ISSymbol ActiveX control element is used for both the Internet Explorer-based and Secure Viewer-based browsers.
Web Tunneling Gateway	The (Primary) Web Tunneling Gateway is a bridge between the web server and the data server that is used in one of two situations. The first is whenever data security is required (e.g., "WinStudio" data exchanged with the Thin Client needs to be encoded). In the second situation, the data server is "hidden" behind a corporate firewall and only the web server IP address (or URL) is displayed. "WinStudio" supports a backup (secondary) Web Tunneling Gateway to be used if the primary Web Tunneling Gateway

Provision as a Web Application

becomes unavailable. The Thin Client automatically switches over to the secondary Web Tunneling Gateway.

The Web Tunneling Gateway is automatically installed when "Rexroth WinStudio" is installed on the PC if the installation program detects that IIS is available.



- The Web Tunneling Gateway is automatically installed if IIS is detected during the installation process. Otherwise, it has to be installed manually.
- The main function of the Web Tunneling Gateway is to encapsulate data packages in HTTP or HTTPS for communication through a firewall.

15.4 ISSymbol Control Layer

ISSymbol is a component designed by "Bosch Rexroth" to display the screens created with "WinStudio" in the web browser and exchange data (tag values and history data) with the TCP/IP server of the project. On the Thin Client station, the web browser (e.g., Internet Explorer) is the container that hosts the ISSymbol control element.

ISSymbol works as a control layer between the project and the web browser, as the Java Virtual Machine for Java-based applications. This approach provides a high level of security because ISSymbol does not allow the project to access the operating system directly.

If the web browser downloads the HTML page specified by the user, it checks for ISSymbol control registration on the current computer. If it cannot be found, the browser tries to download the registry from the URL specified in the project web settings (in the advanced settings of the **Web Settings** in the WinStudio context menu). The web browser is not able to display the screens from the project if the ISSymbol control is not properly registered in the Thin Client station.

NOTICE

Ensure the web browser is enabled to download signed ActiveX control elements in order to automatically download ISSymbol. Otherwise, ISSymbol has to be registered manually in the Thin Client station.

Check the web browser documentation for security settings in case of any questions on how to configure these settings.

Manually installing the ISSymbol control element

The ISSymbol control element can also be installed manually in the Thin Client station. The procedure to install ISSymbol in each operating system is described below:

Windows PC

1. Copy the following files...
 - [...]WinStudio\ Bin\ISSymbolReg.exe
 - [...]WinStudio\ Bin\ISSymbolVM.cab...into any directory of the Thin Client station. Make sure that both files are stored in the same directory.
2. Run **ISSymbolReg.exe** to register the ISSymbol control element on the PC.

Provision as a Web Application

Windows Embedded

1. Determine the OS version and processor type of the Windows Embedded device and find the corresponding redistribution folder in the "WinStudio" program directory:
`[...]\WinStudio\Redist\CE_version\processor_type\Bin\`
 For example, for a MIPS IV processor running Windows CE 5.0 or later:
`[...]\WinStudio\Redist\WinCE 5.0\MIPSIV\Bin\`
2. From that directory, copy the following files...
 - IndHTTP.dll
 - IndVkStd.dll
 - ISSymbolCE.ocx
 ...and **one** of the following Virtual Keyboard layouts...
 - VKEN.ini (for English)
 - VKGE.ini (for German)
 ...into any directory in non-volatile memory on the Windows Embedded device. Make sure that all files are stored in the same directory.
3. In the Windows command prompt, execute the following command:
`regsvrce.exe "\ ISSymbol_directory\ISSymbolCE.ocx"`
 For example:
`regsvrce.exe "\Storage Card\ISSymbolCE.ocx"`
4. Save the registry settings to keep **ISSymbolCE.ocx** registered when re-booting the Windows Embedded device.

Windows Mobile / Pocket PC

1. Follow steps 1 and 2 of the instructions for Windows Embedded (above).
2. Run the program
 "RegSvrCE.exe" on the Windows Mobile device. While in the program, proceed as follows:
 - 2.1 Locate the file "ISSymbolCE.ocx" in the directory to which it has been copied.
 - 2.2 Select the **Register** option.
 - 2.3 Click on **OK**.



The Windows Mobile device may not be supplied with the program "RegSvrCE.exe" as these consumer devices (i.e., smartphones and PDAs) are typically not meant to be used for this type of application. This program is required – or "VBScript.dll", which is also left off on most Windows Mobile devices – please contact the customer support.

Functionality

Open the web browser and enter the URL for one web page available in the server station (e.g., <http://127.0.0.1/main.html>) into the **Address** field. At this point, the Thin Client executes the following process:

1. The web browser downloads the HTML page of the specified screen.
2. The web browser checks for ISSymbol control element registration on the local computer. If it cannot be found, the web browser attempts to download the ISSymbol component from the URL configured in the project (settings saved in the HTML page). Since the ISSymbol control ele-

Provision as a Web Application

ment is properly registered on the Thin Client station, the web browser loads it. From this point on, ISSymbol takes over the communication with the server station, and the web browser is used only as a host for ISSymbol.

3. ISSymbol connects to the data server. The IP address for the data server is configured in the project web settings (**Web Settings** in the WinStudio context menu). This setting is embedded in the HTML page.
4. ISSymbol prompts a window on the Thin Client, requesting the **User Name** and **Password**. The entered data is encoded by [binary control on page 74](#) and sent to the server. The server station checks the validity of the data and whether the rights to open the startup screen are available. If so, the process continues. If not, an error message indicates the User Name and Password are invalid. In this case, the process is not continued.



Step 4 is skipped if the security system is disabled during configuration of the project.

5. ISSymbol downloads the necessary files to display the screen specified by the user (screen files, tags database, translation files, etc.).
6. ISSymbol connects to the data server and reads the value of the tags that are displayed in the specified screen.
7. ISSymbol displays the screen on the web browser and keeps updating the objects according to the values read from the server. Whenever the value of any tag displayed on the open screen(s) changes on the server, the new value is sent to the Thin Client (and vice-versa). Therefore, there is no polling between the Thin Client and the server station. This method increases the communication performance and optimizes the traffic in the network.

Please note that there are two servers in this process:

- **Web server** (HTTP Server): Provides the files from the server to the Thin Client via HTTP protocol over TCP/IP.
- **Data server** (TCP/IP server module from "WinStudio"): Provides tag values and/or history data from the project running on the server to the Thin Client station(s).

Although both servers are usually running on the same computer, "WinStudio" provides the flexibility to run each server in a different station, if necessary. For further information, refer to [Typical architectures for web-based applications on page 399](#).

15.5 Examples of Client/Server Architecture

This section describes some example architectures applied for web-based solutions and provides information on how to configure the project for each architecture. This section does not describe all possible architectures, but it provides the concepts necessary to design and configure different scenarios based on the basic architectures illustrated below

The web settings are configured in the respective configuration dialog. To open this dialog, in the WinStudio context menu, click on **Tools ► Web Settings**. Press the "Advanced" button to access additional settings. The following pictures illustrate these dialogs:

Provision as a Web Application

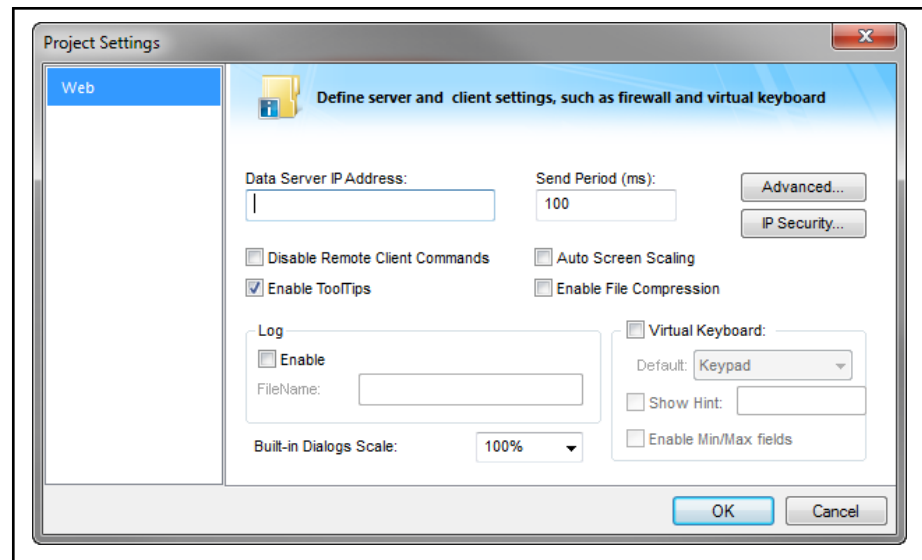


Fig. 15-4: Main web settings (1)

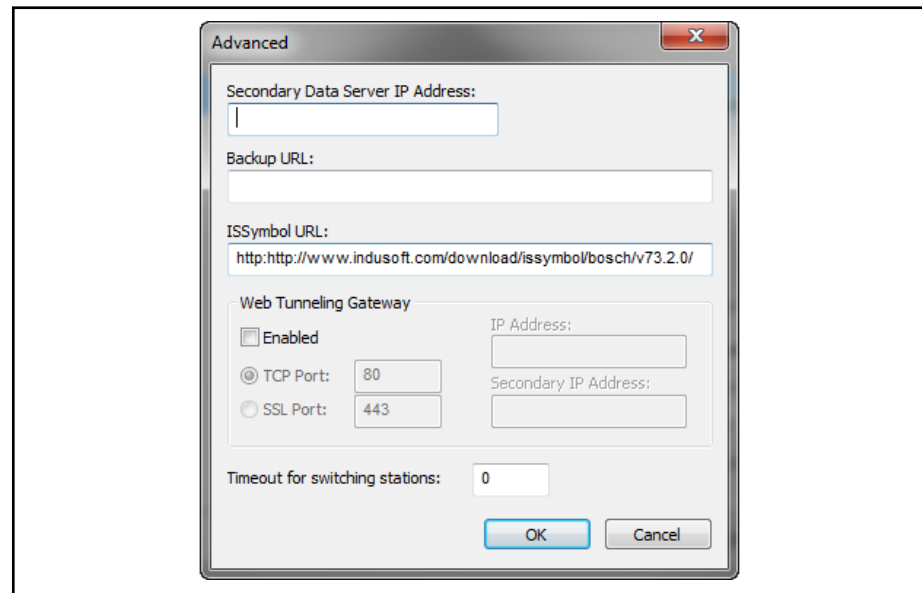


Fig. 15-5: Main web settings (2)

The following table describes the function of the main web settings illustrated in above dialogs:

Setting	Description
Data Server IP Address	If the Web Tunneling Gateway is disabled : The Thin Client control element (ISSymbol) uses the Data Server IP Address to connect to the "WinStudio" TCP/IP server task. If the Web Tunneling Gateway is enabled : The Web Tunneling Gateway uses the Data Server IP Address to connect to the "WinStudio" TCP/IP Server task.
Secondary Data Server IP Address	Same as the Data Server IP Address. However, the Secondary Data Server IP Address is only used if the connection to the Data Server IP Address fails.
Web Tunneling Gateway IP Address	The Thin Client control element (ISSymbol) uses the Web Tunneling Gateway IP Address to connect to the Web Tunneling Gateway.
Web Tunneling Gateway Secondary IP Address	Same as the Web Tunneling Gateway IP Address. However, the Web Tunneling Gateway Secondary IP Address is used only when the connection with the Web Tunneling Gateway IP Address fails.

Provision as a Web Application

The secondary addresses can be used in the following scenarios:

- If the Thin Clients can connect to either one of two redundant servers (web or data); or
- If the Thin Clients can connect to the server via the Intranet (LAN – Local Area Network) or via the Internet (WAN – Wide Area Network). In this case, the primary addresses are configured based on the network used more often by the Thin Clients. In the following examples, the LAN addresses are used as primary and the WAN addresses as secondary addresses.

The following table describes the meaning of some terms used in the following examples:

Term	Description
LAN	Local Area Network (e.g., Intranet)
WAN	Wide Area Network (e.g., Internet)
Server	Station on which the following components are running: Although "WinStudio" does not need to run on the same station as the other components, the following examples are based on that assumption. • "WinStudio" (TCP/IP Server task) • Web server (e.g., Internet Information Services from Microsoft – IIS) • Web Tunneling Gateway for IIS (if enabled)
Thin Client LAN	Thin Client station (web browser + ISSymbol control element) that connects the server via LAN.
Thin Client WAN	Thin Client station (web browser + ISSymbol control element) that connects the server via WAN.
IP_SERVER_LAN	IP Address of the server in LAN.
IP_SERVER_WAN	IP Address of the server in WAN.
IP_ROUTER_LAN	IP address of the router in LAN.
IP_ROUTER_WAN	IP address of the router in WAN.
ScreenName	Name of the project screen, saved as HTML, that is open on the Thin Client station.

Example 1: Web server and Thin Client in the same Intranet (LAN)

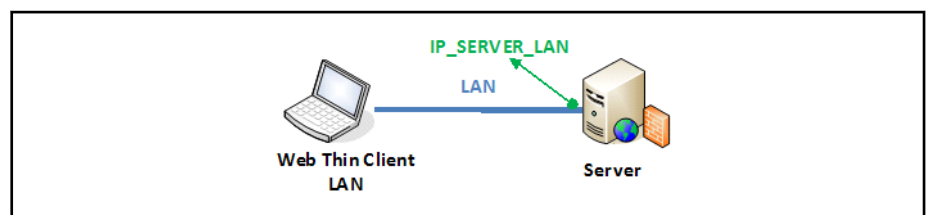


Fig. 15-6: Web server and Thin Client in the same Intranet (LAN)

This is the most common architecture and the easiest to configure. In this architecture, both the web server (e.g., Microsoft IIS) and the data server (i.e., the "WinStudio" TCP/IP server module) are running on the same PC. The Thin Client connects to the web server to download the HTML screen file(s). Afterwards, it connects to the data server to exchange data with the "WinStudio" runtime project. Since both the Thin Client and the server station are connected to the same network, the Thin Client can access the server station directly via its IP address (or host name).

Configuration:

Provision as a Web Application

Setting	Web Gateway enabled	Web Gateway disabled
Data Server IP Address	IP_SERVER_LAN	IP_SERVER_LAN
Secondary Data Server IP Address	-	-
Web Tunneling Gateway IP Address	IP_SERVER_LAN	-
Web Tunneling Gateway Secondary IP Address	-	-



- URL of Thin Client LAN: **http://IP_SERVER_LAN/Screen-Name.html**
- Root directory of the web server (HTTP server) on the server station: "Web" subfolder of the project

Example 2: Web server with Intranet (LAN) and Internet (WAN) connections

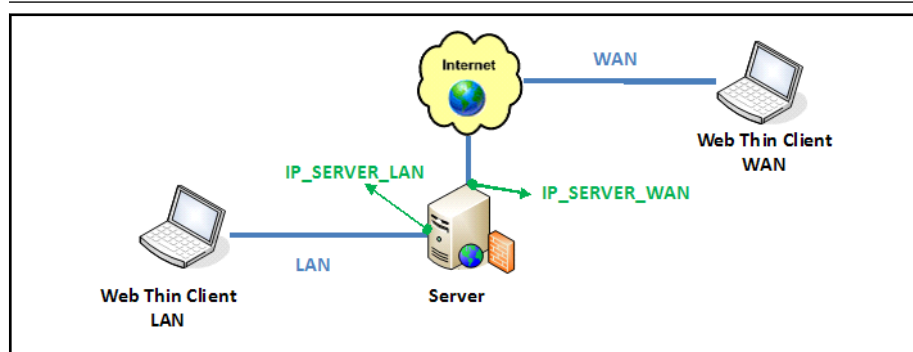


Fig. 15-7: Web server with Intranet (LAN) and Internet (WAN) connections

In this architecture, both the web server (e.g., Microsoft IIS) and the data server (i.e., the "WinStudio" TCP/IP server module) are running on the same PC. Thin Clients can connect to the server either via an Intranet (LAN) connection to the server or an Internet (WAN) connection to the server (e.g., two different Ethernet ports).

Configuration:

Setting	Web Gateway enabled	Web Gateway disabled
Data Server IP Address	IP_SERVER_LAN	IP_SERVER_LAN
Secondary Data Server IP Address	IP_SERVER_LAN	IP_SERVER_WAN
Web Tunneling Gateway IP Address	IP_SERVER_LAN	-
Web Tunneling Gateway Secondary IP Address	IP_SERVER_WAN	-



- URL of Thin Client LAN: **http://IP_SERVER_LAN/Screen-Name.html**
- URL of Thin Client WAN: **http://IP_SERVER_WAN/Screen-Name.html**
- Root directory of the web server (HTTP server) on the server station: "Web" subfolder of the project folder
- A fixed IP address has to be assigned to the web server on the Internet (WAN) and the project has to be running on this server. Consult the ISP provider or IT department for further information about how to get a fixed IP address for the server.

Provision as a Web Application

Example 3: Web server with Intranet (LAN) and router Internet (WAN) connections

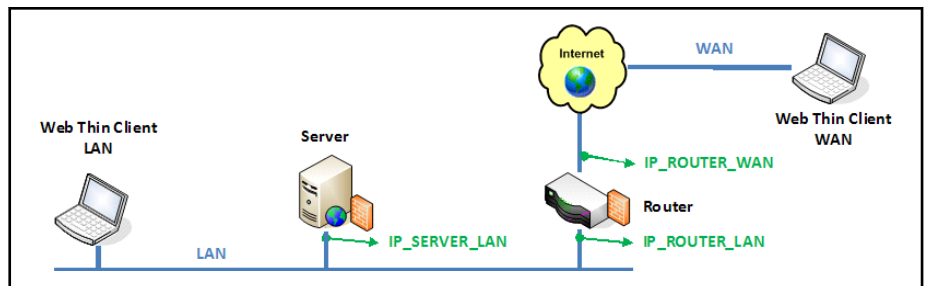


Fig. 15-8: Web server with Intranet (LAN) and router Internet (WAN) connections

In this architecture, both the web server (e.g., Microsoft IIS) and the data server (i.e., the "WinStudio" TCP/IP server module) are running on the same PC. Thin Clients can connect to the server either via an Intranet (LAN) connection or an Internet (WAN) connection. There is a router between the Intranet (LAN) and the Internet (WAN).

Configuration:

Setting	Web Gateway enabled	Web Gateway disabled
Data Server IP Address	IP_SERVER_LAN	IP_SERVER_LAN
Secondary Data Server IP Address	IP_SERVER_LAN	IP_ROUTER_WAN
Web Tunneling Gateway IP Address	IP_SERVER_LAN	-
Web Tunneling Gateway Secondary IP Address	IP_ROUTER_WAN	-



- URL of Thin Client LAN: **http://IP_SERVER_LAN/Screen-Name.html**
- URL of Thin Client WAN: **http://IP_SERVER_WAN/Screen-Name.html**
- The Router has to be configured to forward the TCP Port(s) from its public IP (IP_ROUTER_WAN) to the private server IP (IP_SERVER_LAN).

If the Web Gateway is **enabled**, only the HTTP Port (80, by default) or the HTTPS Port (SSL Port 443, by default) have to be forwarded from IP_ROUTER_WAN to the IP_SERVER_LAN.

If the Web Gateway is **disabled**, both the HTTP Port (80, by default) and the Studio TCP/IP Server Port (1234, by default) have to be forwarded from IP_ROUTER_WAN to the IP_SERVER_LAN. Consult the router documentation for further information about how to configure Port Forwarding on it.

- Root directory of the web server (HTTP server) on the server station: "Web" subfolder of the project folder
- A fixed IP address has to be assigned to the router on the Internet (WAN) and the project has to be running on this server. Consult the ISP provider or IT department for further information about how to get a fixed IP address for the server.

15.6 Configuring the Data Server

"WinStudio" has a couple of dialogs that are used for configuration of the data server and the web server. This information is embedded in the HTML web

Provision as a Web Application

pages of the respective screens. The data server is part of the "WinStudio" runtime project and uses the TCP/IP Server module. There are three (3) basic steps to this configuration:

1st Step: Web Settings configuration dialog

1. In the WinStudio context menu, select **Tools ► Web Settings**.

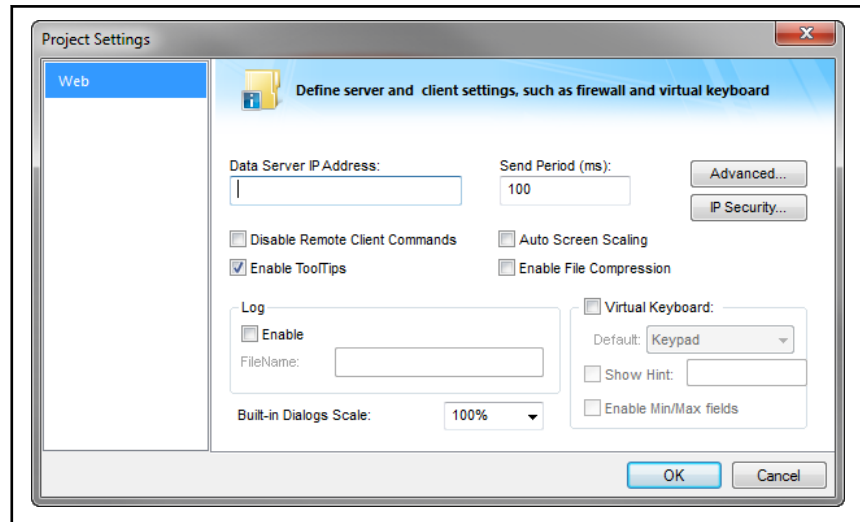


Fig. 15-9: Web Settings configuration dialog

2. In the **Data Server IP Address** field, enter the IP address of the PC where the runtime project is to be created.
3. Make sure that the **Auto Screen Scaling**, **Enable File Compression** and **Enable Tooltips** checkboxes are enabled. If the web client is not to issue commands to the server, enable the **Disable Remote Client Commands** checkbox.
4. If the web client is to use a virtual keyboard, enable the **Virtual Keyboard** checkbox and any additional settings for the virtual keyboard at the Thin Client.
5. Click on the **Advanced** button to define a backup URL (i.e., backup web-site for web pages) and a secondary IP address for the data server. This applies for redundant web servers and/or redundant data servers. Web tunneling can also be enabled in this dialog. The URL for the ISSymbol OCX can also be defined if it is not found on the web server.

Provision as a Web Application

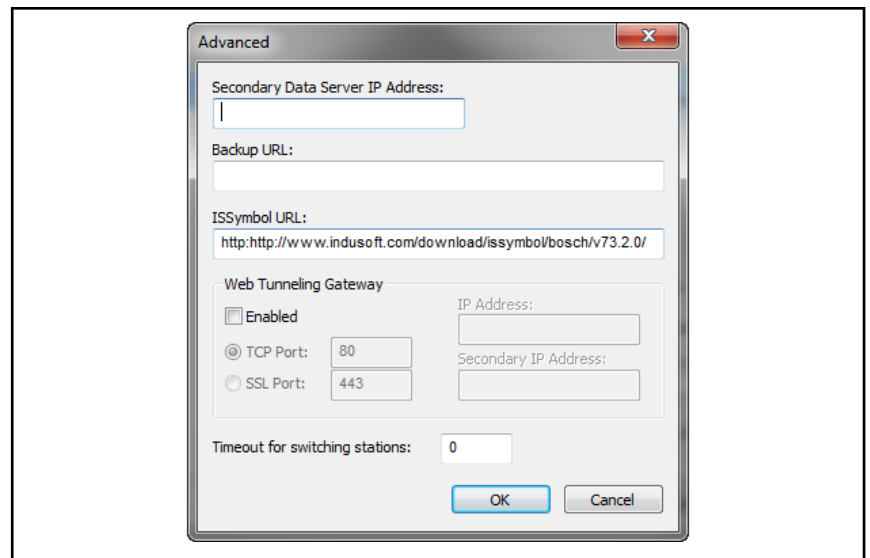


Fig. 15-10: Web Settings configuration dialog, Advanced

6. Click on the **IP Security** button to define a range of IP addresses that are to be accepted by the data server. If a data request is sent from an IP address outside of the defined ranges, the request is not acknowledged. This is an Embedded Firewall function.

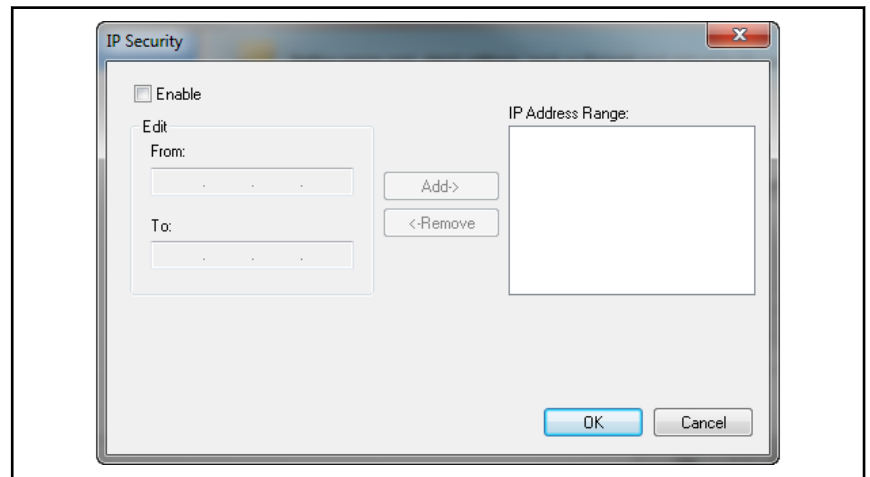


Fig. 15-11: IP Security

2nd Step: Communication settings configuration dialog

1. Open the WinStudio Project Settings dialog and click on the **Communication** tab.

Provision as a Web Application

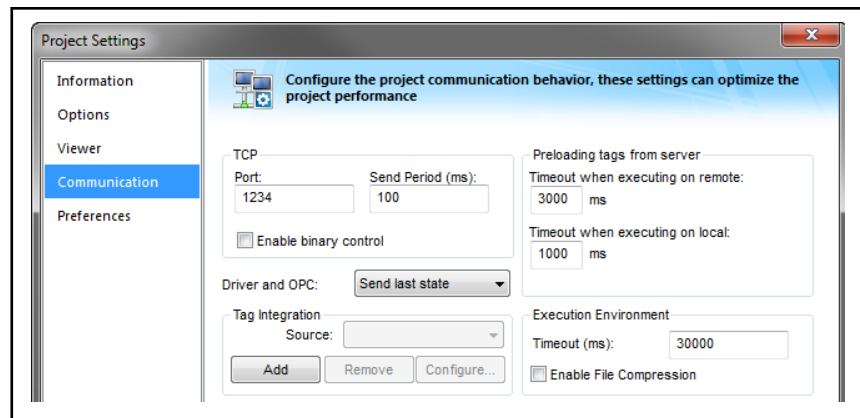


Fig. 15-12: Communication settings configuration dialog

2. Enter the Port number (1234 is the default) for the data server. The Data Send Period can also be defined (i.e., time period for updated communication of data values to the web client).
 3. Optionally, enable **binary control** of the data. This option is more secure, however, slower. The default setting is disabled.
- 3rd Step: Enable the TCP/IP task**
1. Open the task configuration by selecting **Tools ► Status** in the WinStudio context menu.

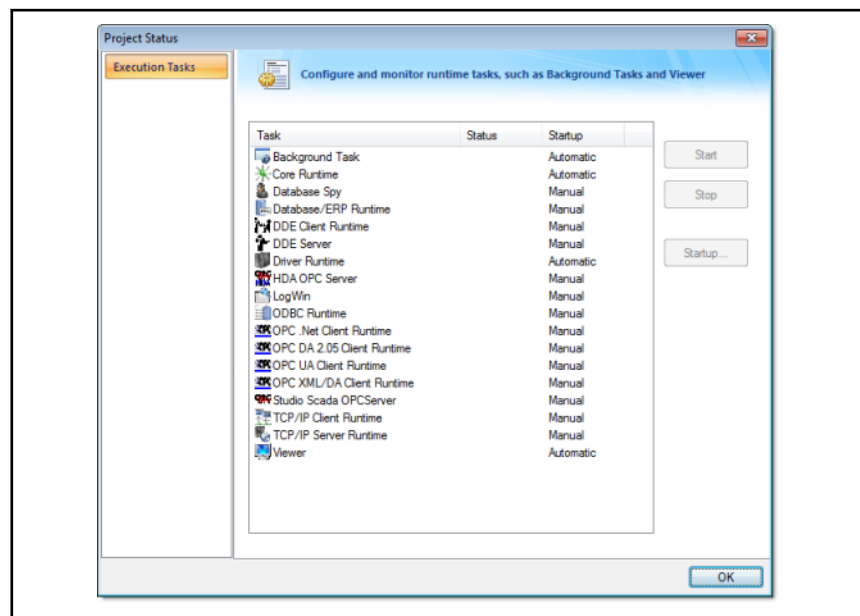


Fig. 15-13: Enable the TCP/IP task

2. Ensure that the TCP/IP server is set to Automatic. This setting is to be default, however, it can be configured manually by clicking on the **Start-up** button.
3. Ensure that the TCP Port number is properly set (see communication settings above). Otherwise, the TCP/IP server starts and stops immediately.



- This configuration information (from steps 1 and 2) is embedded in the web pages generated in the "WinStudio" development environment when selecting **Save As HTML**.
- If any of these settings are changed, the **Verify Project** tool **HAS TO BE** run.

Provision as a Web Application

Using the SetWebConfig function

The integrated **SetWebConfig()** function allows the developer to programmatically configure the data server and web client configuration. The resulting configuration settings are automatically updated in the HTML files of the project (located in the "Web" subfolder). This function always runs on the "WinStudio" server and works in Windows as well as Windows Embedded environments.

For further information, refer to the function [SetWebConfig on page 631](#).

Configuring the web server to support IE and Secure Viewer

"WinStudio" is capable of supporting both Internet Explorer (IE)-based browsers and Secure Viewer-based browsers at the same time. However, the system has to be configured very carefully.

On the web server side

- All HTML files go into the project "Web" subfolder.
- Configure the web server (e.g., IIS) to have its root folder point to the project folder.
- If no Internet connection can be established to the "Bosch Rexroth" website and if the ISSymbol ActiveX control element is not installed on the IE-based web client, save the ISSymbolVM.cab file into the project folder.

On the web client side

- The Secure Viewer-based browser is to be configured to point to the .APP file in the project folder.
- The IE-based browser requires an initial URL as follows:
http://192.168.1.100/Web/startup.html
...where startup.html is the startup web page.

15.7 Configuring a Web Server as Host for Project Pages

As part of providing the "WinStudio" project over the web, a web server has to be configured to host the project screens.

It is not urgently required to use a Windows PC or Windows Embedded device to host the project pages. These pages are essentially static files waiting to be downloaded; runtime processing is entirely handled by the project viewer (i.e., Internet Explorer with [ISSymbol on page 397](#) or Secure Viewer) on the Thin Client. As such, all standard web servers can be used on any computer platform as host for pages.

For example, if a Unix-based intranet server is already available, the project "Web" subfolder (or whatever folder in which the project pages have been saved) can be copied to the server to have the Thin Client point to that server address.

However, please note that the selected web server may not be robust enough to serve as server for the project in a production environment and/or it may not support all "Rexroth WinStudio" functions. If these functions are required, Microsoft IIS is to be used in most cases as described below. In detail:

- To support <Mobile access> (SMA), the server has to be able to process Collaborative Data Objects (CDO) and Active Server Pages (ASP); and
- To support [Web Tunneling Gateway on page 410](#) (WTG), the server has to allow Internet Service API (ISAPI) extensions.

Before installation and configuration of any software, please review its documentation carefully.

Provision as a Web Application



Web servers typically work or "listen" on the TCP/IP port 80 of a computer. Only one running process can receive a defined port. Respectively, if other processes, e.g., SCADA third-party software, is already receiving on port 80, this process might collide with the server process. Either, one of the processes has to be configured to receive from a different port or the Task Manager has to be used to end the conflicting process. If the conflicting process cannot be identified, open the Windows command prompt and enter the following command to get a list of all networking processes:

```
netstat -a -o
```

NTWebServer and CEWebServer

NTWebServers and CEWebServers are lightweight, zero-configuration web servers that are integrated in "Rexroth WinStudio" for free. They can be configured to demonstrate the project and run basic tests without making the financial and technical investment in a full-featured web server. However, they are not robust and do not support SMA or WTG. For proper applications, it is recommended to use Microsoft IIS as described below.

NTWebServer can be executed on any [supported on page 37](#) Windows PC. It can be found in the "WinStudio" program directory under

[...]WinStudio\Bin\NTWebServer.exe.

Copy it to the "Web" subfolder of the project before running it. The server has to be located in the same folder as the pages it is to serve. After running it, this folder becomes the top level (or "home" directory) of the website.

CEWebServers can be executed on any [supported on page 37](#) Windows Embedded device. It can be found in the "WinStudio" program directory under

[...]WinStudio\Redist\CE_version\CE_processor\CEWebServer.exe.

Assuming the "Web" subfolder of the project was copied to the non-volatile memory of the device, copy **CEWebServer.exe** to the same folder and execute it. Again, this folder becomes the top level of the website.



Both NTWebServers and CEWebServers have to run as normal programs as they cannot run as Windows services.

Microsoft IIS

Internet Information Services (IIS) is the full-featured server software that is bundled with Windows Server and "professional" versions of Windows:

IIS versions

Windows version	IIS version	Notes
Windows XP Professional	IIS 5.1	Maximum of 10 simultaneous connections
Windows Server 2003 Windows XP Professional x64 Edition	IIS 6.0	No limit on connections
Windows Server 2008 Windows Vista (most versions)	IIS 7.0	No limit on connections
Windows Server 2008 R2 Windows 7 (all versions)	IIS 7.5	No limit on connections
Windows CE and Windows Mobile 5.0 and later	IIS for CE .NET	Has to be included in Platform Builder Default is a maximum of 10 simultaneous connections

Provision as a Web Application

IIS supports all features of "Rexroth WinStudio" and is robust enough to serve almost any "WinStudio" project in a production environment. This web server is recommended for most users. However, to properly install and configure it, experience in Windows network administration is required.

For reasons of system security, IIS is turned off by default if the operating system is installed. It can be activated in the "Windows Features" dialog (in Windows 7, **Control Panel** ► **Program** ► **Programs and Features** ► **Turn Windows features on or off**):

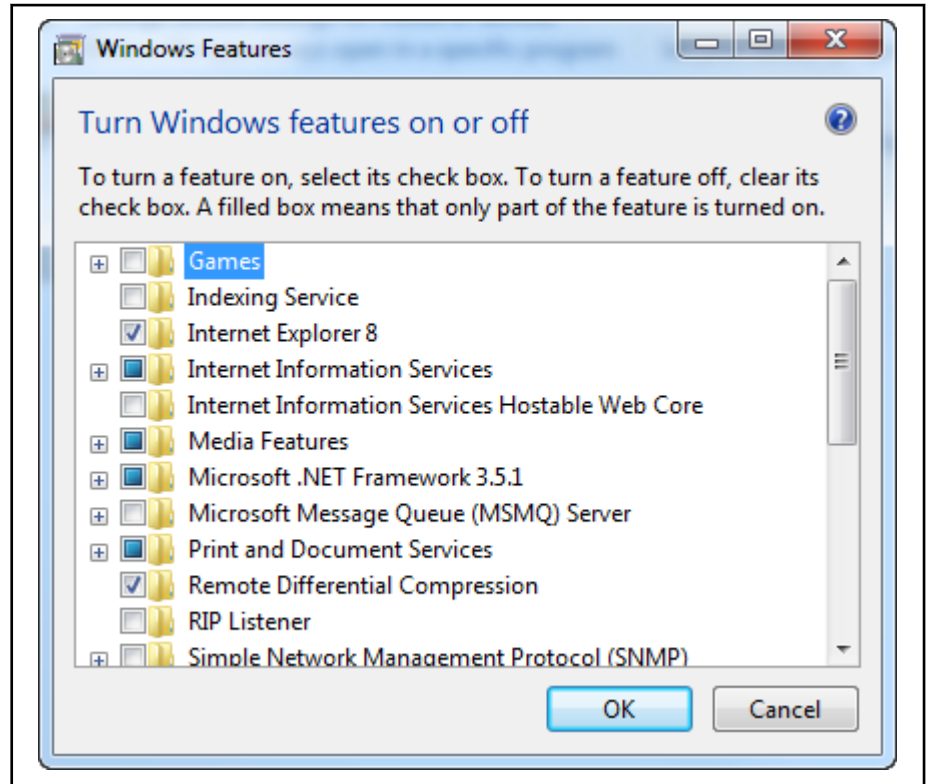


Fig. 15-14: Turning on IIS in the "Windows Features" dialog

If SMA and/or WTG is to be used in the "WinStudio" project, make sure that the ASP and ISAPI Extensions are also activated:

Provision as a Web Application

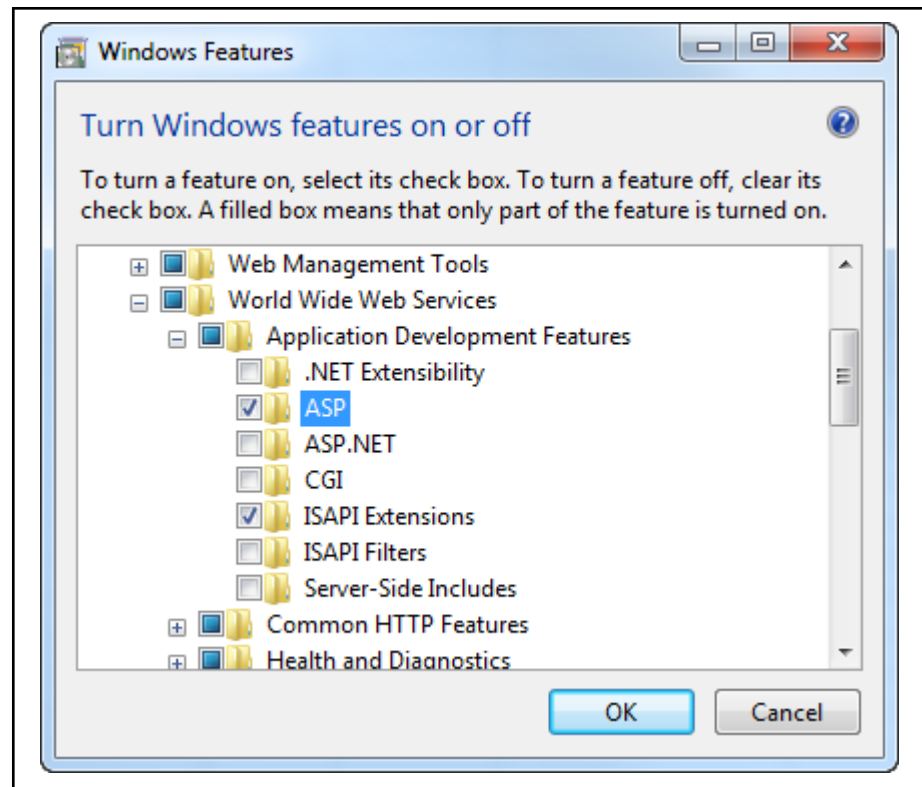


Fig. 15-15: Activating ASP and ISAPI Extensions in the "Windows Features" dialog

If IIS is activated, Administrative Tools (in Windows 7, **Control Panel ► System and Security ► Administrative Tools**) can be used to configure it. For further information, please refer to the detailed documentation provided by Microsoft.

Apache for Windows

If IIS is not available or if it is not to be used, the second most popular web server for Windows is the open-source server [Apache](#). However, this server requires even more expertise than IIS for proper installation and configuration. Please carefully read the respective documentation before using this type of server.

15.8 Installing the Web Tunneling Gateway

The "Web Tunneling Gateway" ("WTG", also referred to as "IndWebSvc") enables Internet-connected computers – typically the web server – to route project data between remote Thin Clients and a data server that is "not" connected to the Internet. This section describes the installation process of WTG.

Before proceeding with this task, Microsoft IIS is to be first set up as a web server to host the project pages. WTG can only run on Microsoft IIS. Also, the web server has to be connected to the Internet and requires a static IP address instead of a dynamically assigned IP address. For information on how to obtain a static IP address, please contact the responsible IT department or internet service provider. Finally, the web server and data server both have to be on the same internal TCP/IP network, even if only the web server is connected to the Internet.

To enable Thin Clients access to the project, they have to be able to communicate with both the web server hosting the project pages and the data server actually running the "WinStudio" project. This is not a problem if servers and clients are all on the same internal network or if both of the servers are con-

Provision as a Web Application

nected to the Internet to enable remote Thin Clients to communicate with them.

However, if the data server is not to be connected to the Internet (for network security or topology reasons), the project data has to be routed through another computer that is connected.

WTG provides these routing capabilities. It is an ISAPI extension for Microsoft IIS and typically installed on the same web server hosting the project pages. It encapsulates the data sent between the data server and Thin Clients and ensures project security. Additionally, it can route data simultaneously between multiple data servers.

These instructions apply for:

- Microsoft IIS 7.0 running on Windows Vista and Windows Server 2008; and
- Microsoft IIS 7.5 running on Windows 7 and Windows Server 2008 R2.

1. On the web server (or the Internet-connected computer acting as gateway), ensure that Microsoft IIS is running and ISAPI Extensions are enabled. For further information, refer to [Configuring a web server as host for project pages on page 407](#).
2. Copy the WTG installer from the "WinStudio" application directory to the "root" directory of the website (e.g., "C:\inetpub\wwwroot\").

The installer is located at

..\WinStudio\Bin\WebTunnelingGateway.exe

3. Execute the installer and follow the steps of the installation wizard. Options cannot be changed. The installer creates a new directory in the "root" directory of the website and installs the WTG extension files in that directory. It also creates a new application pool.
4. Open the IIS Manager. The "Internet Information Services (IIS) Manager" window is displayed.
 - 4.1 Click **Start** and then **Control Panel**.
 - 4.2 In the Control Panel window, proceed as follows.
 - Under Windows Server 2008, click on **Administrative Tools**.
 - Under Windows Vista, click on **System and Maintenance** and then **Administrative Tools**.
 - Under Windows 7 or Windows Server 2008 R2, click on **System and Security** and then **Administrative Tools**.
 - 4.3 In the Administrative Tools window, proceed as follows.
 - Under Windows Vista or Windows 7, double-click on **Internet Information Services (IIS) Manager**.
 - Under Windows Server 2008 or Windows Server 2008 R2, double-click on **Server Manager** and click on **Internet Information Services (IIS) Manager** in the Server Manager window.

The Internet Information Services (IIS) Manager window is displayed.

5. Ensure that the application pool has been created by the installer. If "IndWebSvcPool" is not in the list, the WTG installer did not finish correctly. Return to the beginning of this procedure.
 - 5.1 In the **Connections** pane, expand the server node and click on **Application Pools**.

Provision as a Web Application

- 5.2 On the Application Pools page, in the list of pools, look for IndWebSvcPool.
If IndWebSvcPool is not in the list, then WTG installer did not finish correctly. Return to the beginning of this procedure.
6. Add an ISAPI restriction for IndWebSvc.
 - 6.1 In the **Connections** pane, click on the server node.
 - 6.2 On the **Home** page of the server node, double-click on **ISAPI and CGI Restrictions**.
 - 6.3 On the **ISAPI and CGI Restrictions** page, in the **Actions** pane, click on **Add**.
The Add ISAPI or CGI Restriction dialog is displayed.
 - 6.4 In the **ISAPI or CGI path** field, enter the path of the file **IndWebSvc.dll**
(e.g., **C:\inetpub\wwwroot\indwebsvc\IndWebSvc.dll**), or click on the browse button (...) and navigate to the file.
 - 6.5 In the **Description** field, enter IndWebSvc.
 - 6.6 Select **Allow extension path to execute**.
 - 6.7 Click on **OK**.
7. Configure the handler mappings of the website to make ISAPI extension files executable.
 - 7.1 In the **Connections** pane, expand **Sites**, and click on the website that is acting as gateway (i.e., typically **Default website**, or any site hosting the WinStudio project pages).
 - 7.2 On the **Home** page of the website, double-click on **Handler Mappings**.
 - 7.3 On the **Handler Mappings** page, select **ISAPI-dll** in the list of mappings.
This item is disabled by default.
 - 7.4 In the Actions pane, click on **Edit Feature Permissions**.
The Edit Feature Permissions dialog is displayed.
 - 7.5 Select **Execute**.
 - 7.6 Click on **OK**.
8. Make sure the website and the gateway extension are running and start Internet Explorer on the server and go to <http://127.0.0.1/indwebsvc/indwebsvc.dll>
127.0.0.1 is the "localhost" or "loopback" address, i.e., the browser is displaying the website that is running on the same computer.
The resulting web page generally looks as follows:
Studio Web Gateway
Status: Ok
Version: 1.3
Active Connections: 0
If this is not displayed, either the website is not running or the gateway extension was not correctly configured. Return to the beginning of this procedure.
9. Close the IIS Manager.
10. Configure the "WinStudio" project to include the address of the gateway.

For further information, refer to [Project Settings: Web](#) on page 78.

15.9 Configuring the Thin Client

"WinStudio" supports two types of web browsers to use in combination with Thin Clients: Microsoft Internet Explorer, supporting ActiveX control elements and Secure Viewer developed by "Bosch Rexroth". Both browsers can be used simultaneously with the web server and the choice of browser depends on the project requirements. The Secure Viewer is designed for a production environment, while Internet Explorer is generally more appropriate for desktop applications or where Thin Clients need to connect to different project servers.

Configuring Microsoft Internet Explorer

The configuration of Microsoft Internet Explorer to use as web browser on a Thin Client includes only a few steps. The most important elements include:

- Ensure that ActiveX control elements are enabled. In general, the required settings can be found under **Tools ► Internet Options ► Security**. Click on the **Custom Level** button and ensure that the ISSymbol ActiveX control can be installed. Afterwards, reset a higher security level. (Otherwise, be sure to install the ISSymbol ActiveX control manually).
- Ensure that VBScript is supported on the PC. This is generally not an issue with desktop PCs, however, it can be an issue on Windows Embedded devices. It has to be ensured that the VBRUN.dll is installed and registered on the PC.

If Microsoft Internet Explorer is to be used as web browser for Thin Client projects, individual navigation bars and buttons can be removed to reduce the amount of space taken up by these items.

On the desktop, an icon can be created as shortcut to the startup web page on the web server. Click this icon to access the web page directly. This shortcut can also be placed in the Startup folder to start the web page automatically whenever the browser is started.

Installing Secure Viewer

Installation on a Windows PC

Secure Viewer has to be started from CD (or downloaded from the "Bosch Rexroth" website). Proceed as follows:

1. Request the Secure Viewer installer from the software vendor, save it to the designated computer and start it.
2. Follow the instructions of the installation wizard. There are only two settings that have to be configured during installation:
 - **URL:** Enter the URL or file path of the project file (**project_name.app**) on the web server.
 - **Server IP:** Enter the IP address or hostname of the data server (TCP/IP Server).
3. Finish the installation by clicking on **Finish** to close the installer.

The installation wizard automatically installs and registers the Thin Client Active X component (ISSymbol.ocx). If the **URL** and **Server IP** settings are configured correctly, Secure Viewer is ready to start.



To increase security of the runtime environment, add a Secure Viewer shortcut (Viewer.exe) to the Startup directory in Windows. Viewer.exe is installed into the \Bin subfolder where the "WinStudio" program is installed.

Installing on Windows Embedded

Provision as a Web Application

By copying the necessary files to the non-volatile memory of the device, Secure Viewer can also be installed on a Windows Embedded device. To install Secure Viewer:

1. Determine the OS version and processor type of the Windows Embedded device and browse to the corresponding subfolder in the "WinStudio" program directory. For example: If Windows CE 5.0 is running on a MIPS processor, browse to the [...]WinStudio\Redist\WinCE 5.0\MIPS\Bin subfolder.
2. *Select the following files in the subfolder:*
 - IndVkStd.dll
 - ISSymbolCE.ocx
 - Tagi.bin
 - Viewer.exe
 - ViewerCfg.exe
3. Copy the files to the non-volatile memory on the Windows Embedded device.
4. Register the ISSymbol control on the Windows Embedded device by executing the following command in the **Command Prompt** window:
regsvrce.exe "ISSymbolPath \ISSymbolCE.ocx"
 Example:
regsvrce.exe "Storage Card\ISSymbolCE.ocx"
5. Save the registry settings to keep **ISSymbolCE.ocx** registered when rebooting the Windows Embedded device.



Check the documentation of the device manufacturer for information on how to save the registry settings.

Changing the Secure Viewer configuration

6. Start Secure Viewer by running **Viewer.exe**.

After initial installation, the configuration of Secure Viewer is saved in the **viewer.ini** file that is to be in the same directory as **Viewer.exe**. (This is to be in the \Bin subfolder where the Secure Viewer program is installed). There are two options to change the configuration of Secure Viewer after it has been installed. First, the also installed viewer configuration utility can be run (**ViewerCfg.exe**).

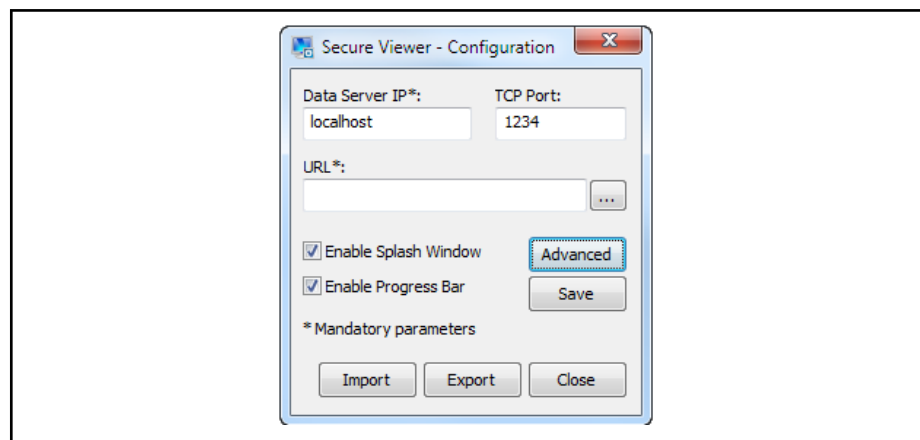


Fig. 15-16: Dialog ViewerCfg

Provision as a Web Application



On Windows Vista and Windows 7, administrator rights are required to properly run the Secure Viewer configuration utility. It is generally installed with these rights by default. In case of any further problems, check the file properties of "ViewerCfg.exe" and ensure that **Run as Administrator** is selected.

The configuration utility provides the following options:

- **Save** button: Immediately saves the current settings to a "viewer.ini" in the same directory as the "ViewerCfg.exe" utility.
- **Import** button: Loads a "viewer.ini" file from another directory.
- **Export** button: Saves the "viewer.ini" file to another directory.
- **Data Server IP** field: Enter the IP address (or host name) of the data server station. The data server station is the computer or device on which the TCP/IP Server module is running.
- **TCP Port** field: Enter the port number of the data server, if it is different than the default port of 1234.
- **URL** field: Enter the URL or file path of the project file ("*.app") on the web server.
- **Enable Splash Window** option: Check (enable) this option to display a splash window when starting Secure Viewer.
- **Enable Progress Bar** option: Enable this option to display a progress bar while the project file is loaded by Secure Viewer.
- **Advanced** button: Click to access additional configuration options:

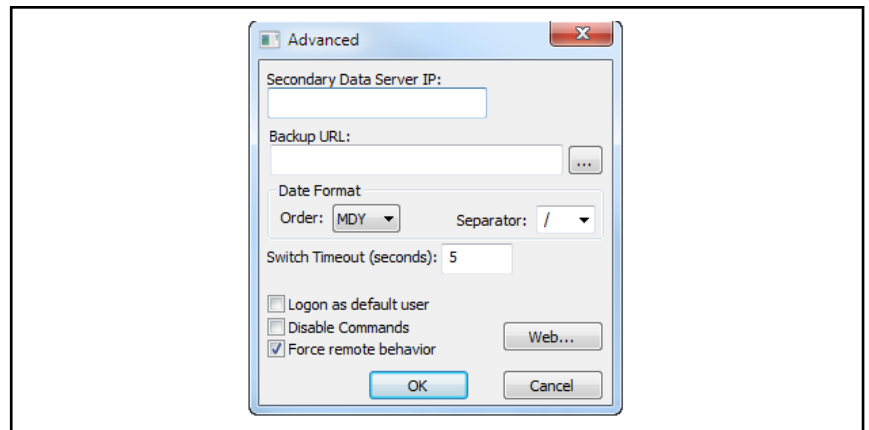


Fig.15-17: Secure Viewer Configuration utility – Advanced

- **Secondary Data Server IP** field: Enter the IP address (or host name) of the secondary data server station. If the primary data server fails, Secure Viewer automatically attempts to connect to the secondary data server.
- **Backup URL** field: Enter the URL of the backup location of the project file.
- **Date Format**: Adjust the date format on the Secure Viewer station to match the data server. This is to maintain compatibility with any Alarm/Event Control or Trend Control objects in the project.
- **Switch Timeout** field: Enter the number of seconds that Secure Viewer is to wait before attempting to connect to the secondary server if the connection to the primary server is interrupted. (For further information, refer to [Web Tab on page 78.](#))

Provision as a Web Application

- **Log on as default user** option: Enable this option to have Secure Viewer automatically log on as Guest, not requiring a username or password.
- **Disable Commands** option: Enable this option to prevent Secure Viewer from sending user commands back to the server. Only current information from the server are displayed.
- **Web** button: If a Web Tunneling Gateway has been configured to connect the Intranet to the Internet, the gateway addresses are to be entered here.

The second option to change the configuration of Secure Viewer is to manually edit the **viewer.ini** file with a text editor. The structure of the file is as follows:

File structure

```
[Options]
nosplash=           // Enable (0) or disable (1) the splash window
noprogressbar=      // Enable (0) or disable (1) the progress bar
ds1=                // Data Server Primary
ds2=                // Data Server Secondary
dsp=                // Data Server Port
wtg1=                // Web Tunneling Gateway Primary
wtg2=                // Web Tunneling Gateway Secondary
url=                // URL from project file (*.app)
proxyip=            // Proxy Address
proxyp=             // Proxy Port
ceemul=             // Enable (1) or disable (0) CView emulation
```

Example

PC with "WinStudio" runtime project (has both web server (NTWebServer or IIS) and data server):

- IP Address is 192.168.1.106
- Project name is SecureViewerTest
- Project file is SecureViewerTest.app

Secure Viewer configuration dialog:

- Data Server IP = 192.168.1.106
- TCP Port = 1234
- URL = http://192.168.1.106//SecureViewerTest.app

Alternatively, the viewer.ini file could be edited as follows:

viewer.ini

```
[Options]
url=http://192.168.1.106//SecureViewerTest.app
noprogressbar=1
ds1=192.168.1.106
nosplash=1
ds2=
dsp=1234
wtg1=
wtg2=
user=Guest
```

```
[OEM]
Splash=Splash.bmp //this is a splash screen
```



- The "Rexroth WinStudio" runtime license has to support a Secure Viewer (Thin Client). Otherwise, the connection to the data server is refused.
 - Ensure that NTWebServer is stored in the project folder (not in the "Web" subfolder) or point IIS to the project folder.
-

15.10 Implementing Security for Web-based Applications

There are various methods for implementing security of web-based applications. The required approach may depend on a number of factors and may involve one or more methods of implementing security.

Method 1: Password protection

In "WinStudio" groups of users and individual users within a group can be created. Each group (e.g., operators, supervisors, maintenance) can have different security levels and access different levels of functionality. Individual passwords can be configured for each user.

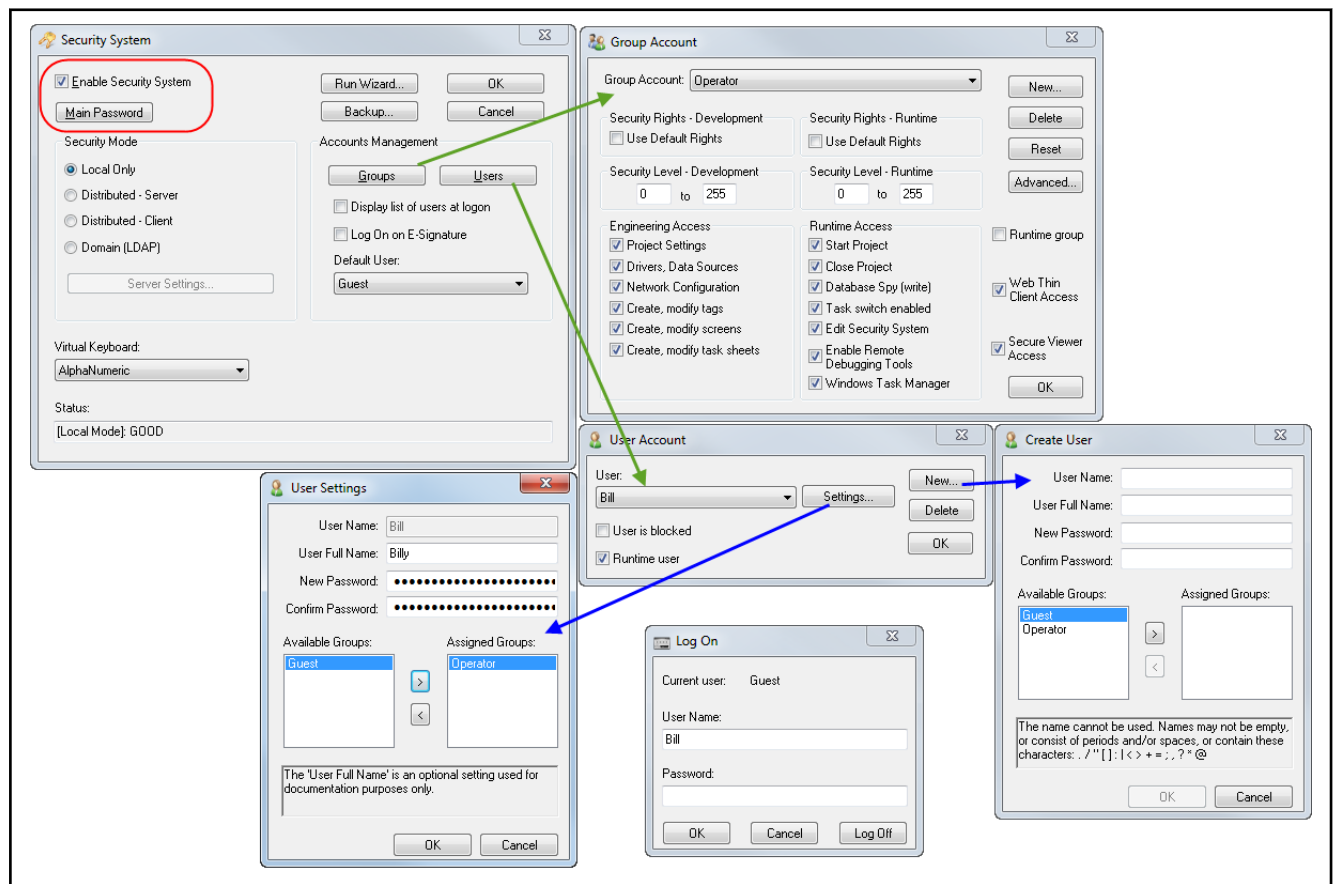


Fig. 15-18: Security groups and users

In addition, groups can be assigned advanced settings, allowing features like minimum password size, password aging, e-signature on objects with command animations, account auto-lockup (e.g., lock up after a number of invalid attempts to access), and user account blocking (temporarily disable – e.g., when employee is on vacation).

Provision as a Web Application

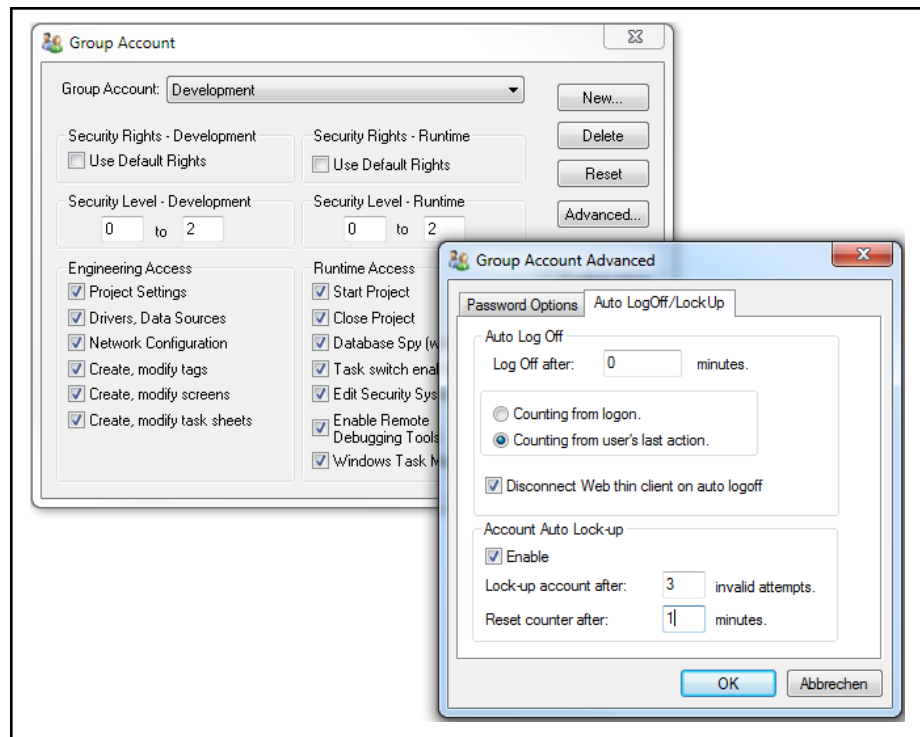


Fig. 15-19: Account auto-lockup and user account blocking

If system security is enabled, these password protection functions are also available at the Thin Client station. Users at Thin Client stations attempting to connect to the web server are required to enter a User Name and Password. If these are invalid, the user cannot access the system.

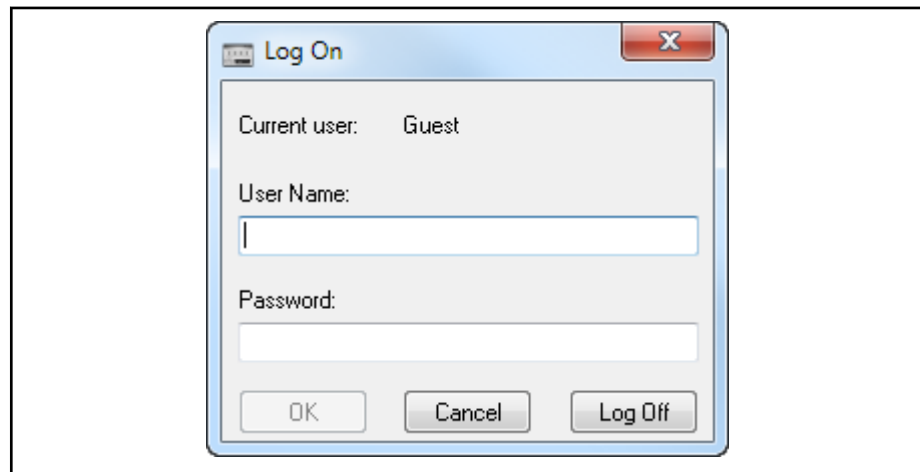


Fig. 15-20: Log On dialog

Within a project, the various screen objects and animations, as well as screen access can be assigned a security level. The current user logged on requires an access level range corresponding to the desired object or screen. The following is a representative method of assigning security access levels by group.

Provision as a Web Application

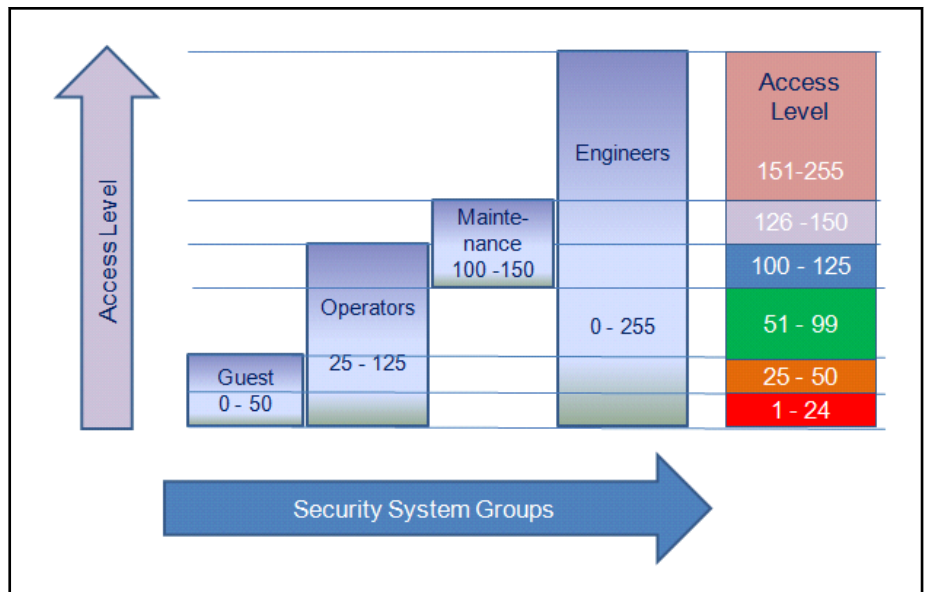


Fig.15-21: Screen access

For further information, refer to [Security on page 357](#).

Method 2: Disabling Thin Client commands

"WinStudio" allows bi-directional data exchange between Thin Client and data server. However, for security reasons it may be advantageous to only allow the Thin Client to view the process or machine data and not send any data back to the data server.

Enable (check) the **Disable Remote Client Commands** option in the Project Settings (in the WinStudio context menu) to ensure that all commands from Thin Client stations are blocked. The communication becomes unidirectional (from server to Thin Clients):

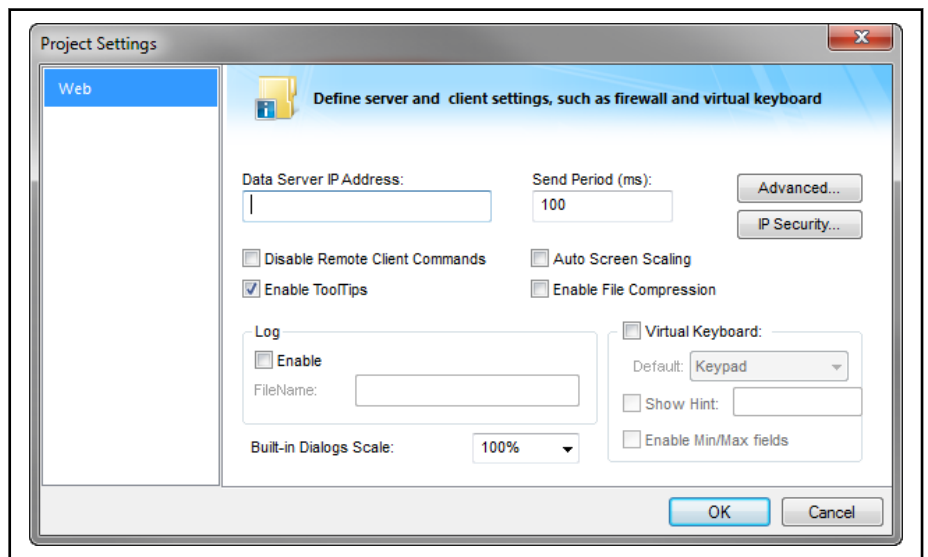


Fig.15-22: Web Settings dialog

Method 3: Embedded Firewall

This function enables filtering the access to the project based on the IP address of the Thin Clients. If a Thin Client attempts to connect to the server station, the server checks if the IP address of the Thin Client station is authorized to access the project. The ranges of authorized IP addresses can be configured by clicking on **IP Security** in the Web settings.

Provision as a Web Application

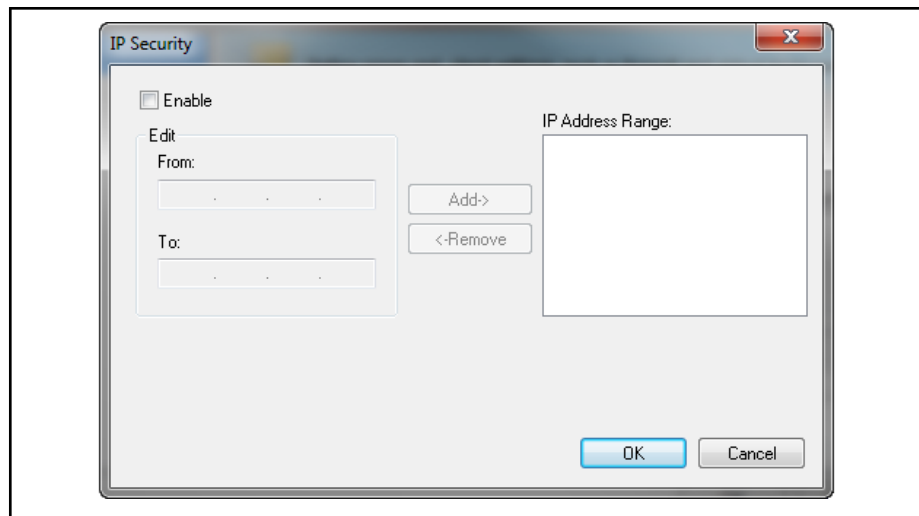


Fig. 15-23: IP Security dialog

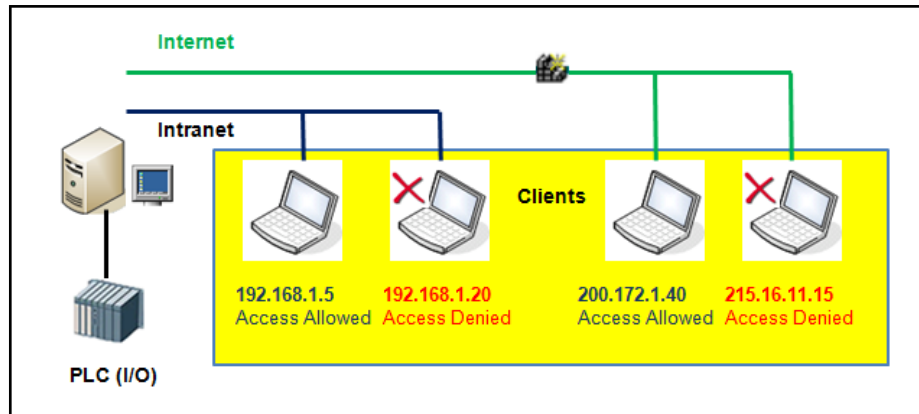


Fig. 15-24: Access allowed by IP address

Method 4: Encoded Communication (SSL)

By enabling the Web Tunneling Gateway (WTG), the entire communication between data server + web server and Thin Client can be encoded using RC6, a highly-secure 128-bit encryption standard. To use SSL, proceed as follows:

1. In the "Web Settings" dialog, click on **Advanced**. Enable the **Web Tunneling Gateway Enabled** option. Click on the **SSL** radio button and ensure that the SSL port is set to 443. Click on **OK**.

Provision as a Web Application

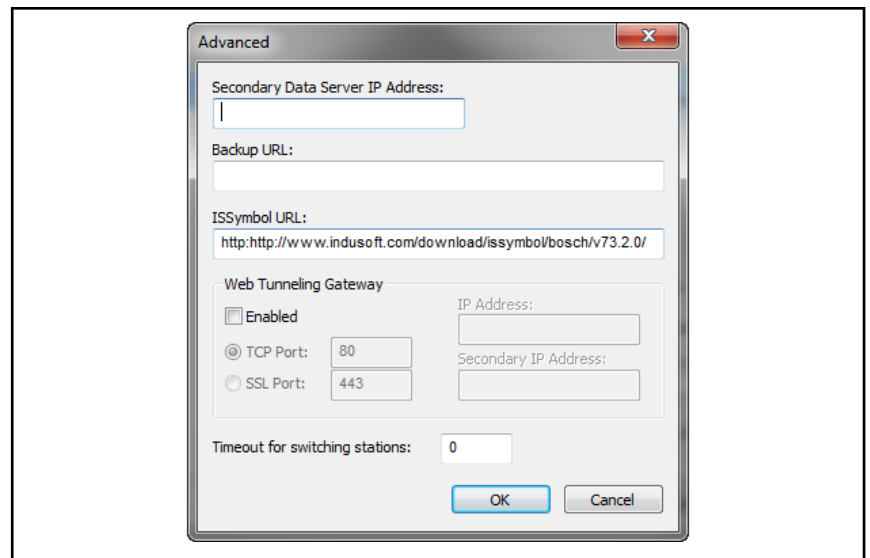


Fig. 15-25: Web Settings – Advanced dialog

2. Ensure that the SSL functions are enabled on the web server and that a SSL certificate of authentication is available.
3. Ensure that SSL is enabled in the Web Client.
4. Set up all additional web configurations to support the WTG.

Method 5: VPN

A VPN is a Virtual Private Network. This type of network is referred to as virtual as it actually uses the public Internet to transport data from one computer to another. However, since this type of network is encoded and uses other security mechanisms enabled by the ISP, it is a very secure private network. While VPNs are inherently secure, they are more costly than a simple public Internet connection.

15.11 Port Usage

There are various ports that are used by "WinStudio" and/or related software. These are:

Port #	Program
20	FTP Server (Data)
21	FTP Server (Command)
25	SMTP Server
80	Microsoft IIS Server for HTTP packets
110	POP3
118	Microsoft SQL Server Services
161	SNMP
162	SNMP Trap
389	LDAP
443	Microsoft IIS Server for HTTPS packets (SSL)
502	Modbus TCP/IP protocol
663	LDAP over SSL
1028	FTP Client (Command)

Provision as a Web Application

Port #	Program
1029	FTP Client (Data)
1234	Project TCP/IP Server
1443	Microsoft SQL Server
1444	Microsoft SQL Server default port (Monitor)
1521	Oracle
1526	Oracle
2030	Oracle
3001	A-B Ethernet TCP/IP Protocol (default)
3306	MySQL (can be configured to use 3306-3309)
3872	Oracle Management Remote Agent
3997	Studio ADO Gateway
4322	Remote Agent (CEServer)
5432	PostgreSQL
47808	BACNet UDP Protocol (default)

You may need to accommodate one or more of these port's usage in your Firewall settings.

15.12 Exercise: Viewing the Project on the Web

To view the project, proceed as follows:

1. Expand the "WinStudio" folder and double-click on the **main.scr** screen.
2. To save the screen in HTML format, click on **Save as HTML** in the context menu of the screen.

Web files are saved in the "Web" folder.

Respectively, open this folder and check whether the "Main" screen has been saved successfully. The file **main.html** is displayed.

3. Open the "Web Settings" dialog.

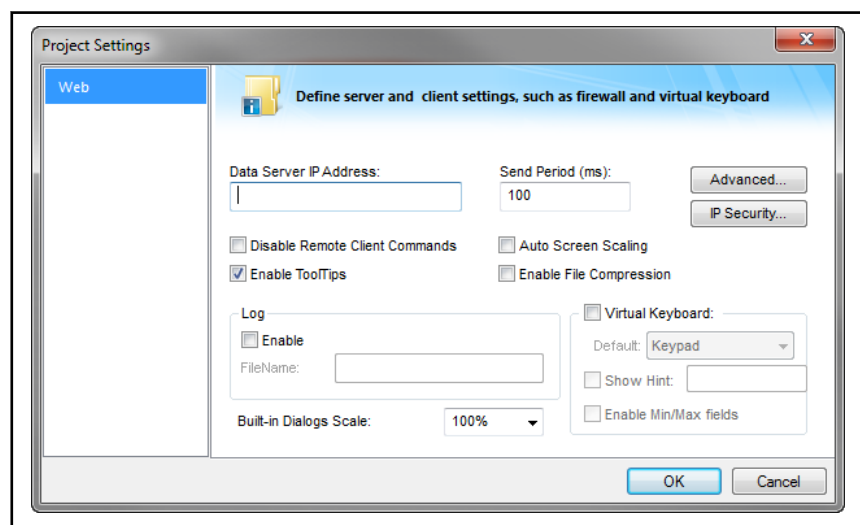


Fig. 15-26: Opening the Web Settings dialog

Provision as a Web Application

4. Configure the **Data Server IP Address** to use the IP address of the server station (computer on which the program is running) at runtime. The Thin Client station exchanges online data (tag values) with the station specified in this field.
5. Enter the URL path of the file **main.html** (in the "Web" folder) into the **URL** field.

The URL depends on the Home directory configured in the web server of the server station.



Microsoft provides web servers for any Microsoft operating system. For information on the installation and configuration of web servers, refer to the Microsoft documentation.

6. After configuring the web settings, click on **OK** to close the dialog.
7. Save and close all screens and worksheets and verify the project (**Verify Application** in the WinStudio context menu) to update the web settings of the web page.

NOTICE

The project has to be verified again after each change to the web settings.

To test the web-based application, proceed as follows:

1. Click on **Start application** (in the "WinStudio" context menu), to execute the project locally on the server station.
2. Open an Internet browser (Microsoft Internet Explorer or Netscape) and enter the URL to open the main.html screen from the server station.
3. If the "Log On" dialog is displayed in the browser, enter **Guest** into the **User Name** field and click on **OK** to open the main.html screen in the browser.

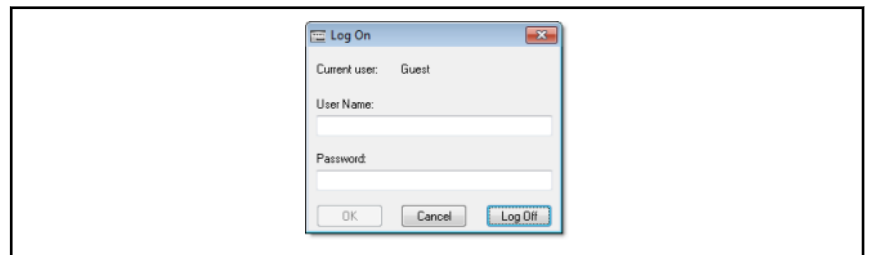


Fig. 15-27: Log On dialog

Please note that the level of any tank can be changed locally (server station) with the Viewer runtime module or remotely (Thin Client) with the browser.



A Thin Client requires an ActiveX component (**ISSymbol.ocx**) to handle screens in the browser. If the Thin Client is connected to the Internet, this component is downloaded and registered automatically.

16 Downloading to a Remote Device

16.1 Configuring the Target System

After configuring a project and testing it locally (on the development workstation), the project can be downloaded to a remote runtime workstation running "WinStudio" on Windows XP, XPe, W7 or CView on a Windows Embedded device.

1. Ensure that the Remote Agent ("CEServer.exe") is running on the target workstation (remote station).
 - On a Windows platform, the "CEServer.exe" file is located in the [...]WinStudio\Bin folder.
 - On a Windows Embedded device, the file is located in the [...]WinStudio folder.
2. Execute "CEServer.exe" on the target workstation and click on "Setup" in the **Remote Agent** dialog.

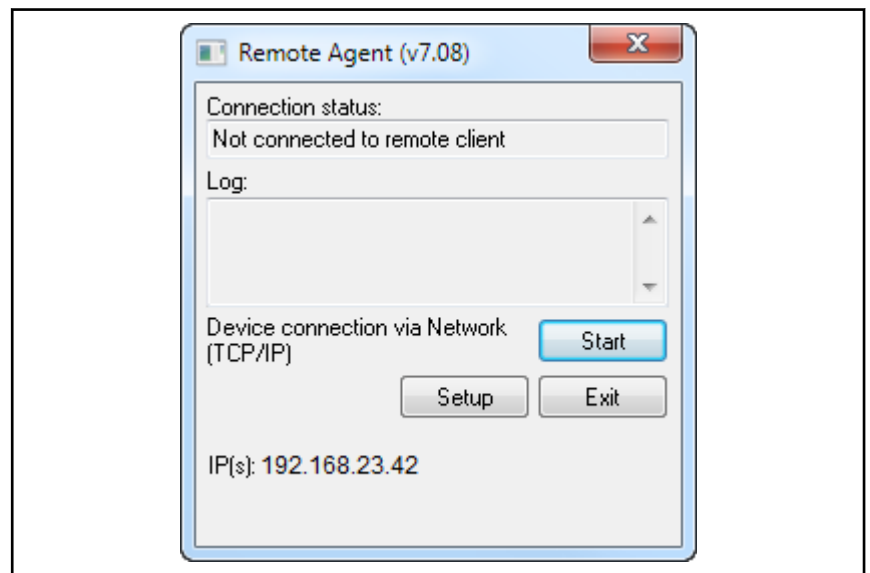


Fig.16-1: Remote Agent dialog

The "Setup" dialog is displayed:

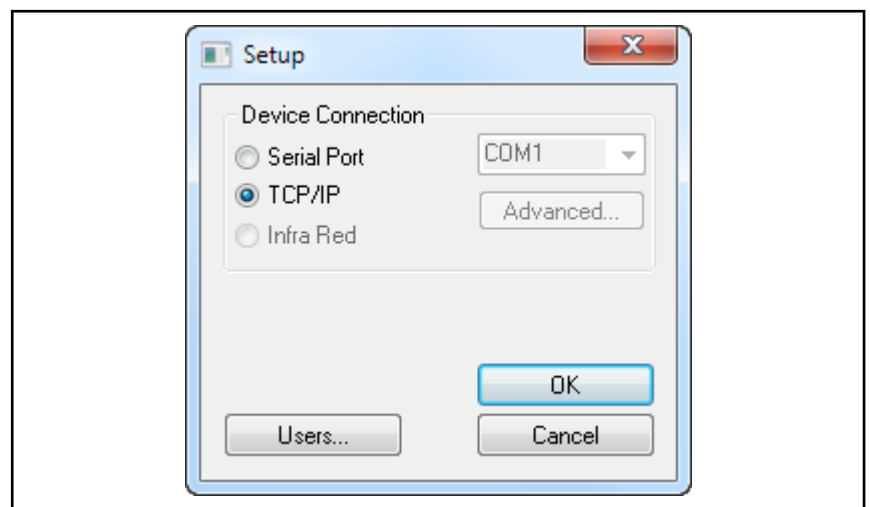


Fig.16-2: Setup dialog

Downloading to a Remote Device

Use the parameters on this dialog to configure communication between the development and target systems:

- **Serial Port:** To establish a serial connection to the project server, select "Serial Port" and then select the specific port from the selection list. Click **Advanced**, to display the "Serial Advanced Settings" dialog.

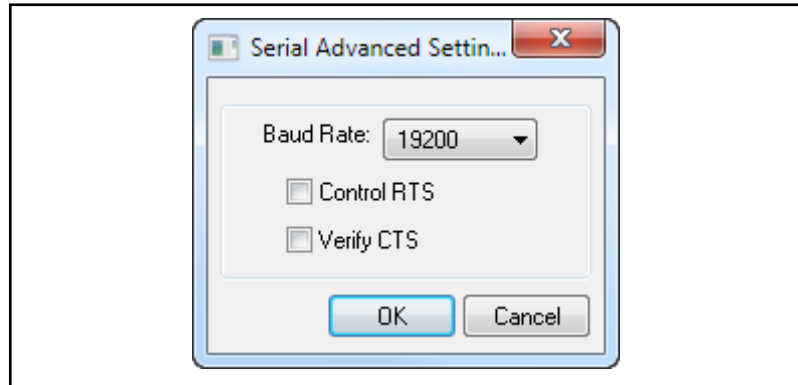


Fig. 16-3: Advanced Settings dialog

The parameters on this dialog can be used to control the flow of data between target and development stations: After adjusting these parameters, click **OK** to close the "Serial Advanced Settings" dialog.

- **TCP/IP:** Enable this option to establish the connection to the development station via TCP/IP.



For better performance, it is recommended to use a TCP/IP connection instead of a Serial Port connection.

3. To close the "Setup" dialog, click on **OK**. Leave the Remote Agent program running in the remote workstation.

Continue with [configuring the development station on page 426](#).

16.2 Configuring the Development Station

After configuration of the target system to receive data, proceed as follows to transfer the project data from the development station to the target device:

1. In the WinStudio context menu, select **WinStudio Target Platform**. The "Remote Management" dialog is displayed.

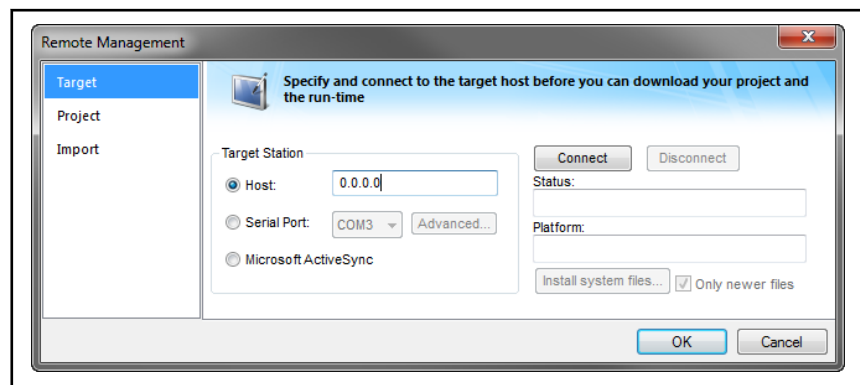


Fig. 16-4: Remote Management dialog

Downloading to a Remote Device

2. Select the **Target** tab and use the following options to specify the target system to connect to.
 - **Host:** Select this option and enter the IP address of the target system into the text box if TCP/IP connection was specified when configuring the target system.
 - **Serial Port:** Select this option and select a port from the combobox if a Serial Port connection was specified when configuring the target system.
 - **Microsoft ActiveSync:** Select this option to connect to devices on which ActiveSync is enabled.
3. Click on **Connect** to connect to the target system.
4. Select the **Project** tab.
5. In the **Target** field, check the directory for the project files on the target system. If the path is incorrect, click on the browse button to the right of the **Target** field to select a new directory. (If a proper connection to the target system is established, it can be browsed like any network drive.)

The project files can only be downloaded to folders that are under the folder in which CEServer is installed and running.

For example, if CEServer is installed at `\StorageCard\CEView\`, the project files are to be installed at `\StorageCard\CEView\<project_name>\`.
6. Click on **Download** to download the entire project to the target system. (Or click on **Send File** to select an individual file to send.)

NOTICE

If a project is sent to the target system, the new project replaces the old one automatically and immediately, even if the old project is still running.

If the Only newer files option is additionally disabled, the development application deletes all of the old files in the project folder before downloading the new files.

In case of doubt the old project may be stopped (by clicking on Stop) before sending the new one. This way, it is ensured that the application is closed properly and not influenced if files are deleted.

However, it is not absolutely necessary to stop the project. If the project is configured in a stable way, new files can be sent at any time.



- After initial configuration of the Remote Management settings, **Send project to target** can be used to send updated project files at any time without opening this dialog.
- If the download is interrupted, the development application requests confirmation to continue and indicates that the project may not run properly.

7. After the download is complete, click **Start** to start the project on the target system.

Downloading to a Remote Device

16.3 Automatically Running a Project

By default, the finished project has to be executed manually on the Windows Embedded device either by connecting to the target device from the development computer using the WinStudio target platform dialog and starting the application there (see above) or at the device itself by clicking the **Start** button in the "Remote Agent" dialog.

However, the Windows Embedded device can be configured to automatically execute a specified project. To do so, edit the file **CEServer.ini** on the Windows CE device to include the following setting:

[Setup]

AppName=Application Path

Where **project_name** is the location of the "WinStudio" project files on the Windows Embedded device.

For example:

[Setup]

AppName=\\StorageCard\\Test\\CEServerTest

The next time the Windows Embedded device boots up and the "Remote Agent" dialog (**CEServer.exe**) is opened, this setting is read out and the specified project is automatically executed.

There are three ways to edit the **CEServer.ini** file:

- Edit the file directly on the Windows Embedded device using an attached keyboard or the touchscreen keypad. The file generally is in the same folder as the file <CEServer.exe>.
- Connect the Windows Embedded device as a shared drive to the PC and edit the file there.
- Edit the file in the [...]WinStudio\\Redist directory "before" installing the system files on the Windows Embedded device.

NOTICE

This last method changes the default copy of CEServer.ini that is provided with "WinStudio".

Use this method only if:

- The files have been saved before editing
- The same system files are installed on multiple, identical Windows Embedded devices and
- The directory (file path) of the "WinStudio" project files on the device is already known (perhaps by using the normal installation method on a test device).

17 Database Interface

17.1 Database Interface, General Information

Configuring a database interface with "WinStudio" is basically linking tasks from "WinStudio" (alarms, events or trends) to tables of external databases via a specific database provider that supports the selected database.

Each history task (alarm, events or trend) can be configured to save data either to files with the proprietary format of "WinStudio" or to external SQL relational databases. Use the Options tab to configure the database to save the alarm and event history. (For instructions on saving the history of trend tasks, refer to [Trend folder on page 269](#).)

"WinStudio" supports ADO.NET to provide an intuitive, simple, flexible and powerful interface with standard technologies from MDAC (Microsoft Data Access Components) such as OLE DB (Object Linking Embedded Database) and ODBC (Open Database Connectivity). By using this capability, a connection can be established to any MDAC compatible database (please refer to the conformance test table for a list of databases already tested).

The following tasks support the database interface:

- **Alarms:** WinStudio can save and/or retrieve alarm history messages in a relational database.
- **Events:** WinStudio can save and/or retrieve event messages in a relational database.
- **Trends:** WinStudio can save and/or retrieve trend history values in a relational database.
- **Viewer:** Database information can be displayed both in table format ([Alarm/Event Control on page 262](#) and [Grid on page 305](#) objects) or in a graphical format ([Trend control on page 277](#) object).
- **Web:** As the items listed in the following are already available in the "WinStudio" web interface, a project can be deployed that stores/saves data in a relational database and sends it over the web.

Using its embedded database interface, "WinStudio" can easily provide data from the production level to third-party systems (e.g., ERP) or receive data from them.

"WinStudio" can communicate with any relational database supported by a valid ADO.NET provider, OLE DB provider or ODBC driver. However, the conformance tests have been executed with the following databases:

Conformance test table

Database	Database version	ADO.NET provider	Assembly version
Microsoft SQL Server 2000	8.0	System.Data.SqlClient	1.0.5000.0
Microsoft Access 2000	9.0.3821 SR-1	System.Data.OleDbClient	1.0.5000.0
Microsoft Excel 2000	9.0.3821 SR-1	System.Data.OleDbClient	1.0.5000.0
Oracle	10g Release 1 for Windows	System.Data.OracleClient	1.0.5000.0
Sybase	Anywhere 9.0.1.1751	iAnywhere.Data.AsacClient	9.0.1.1751
MySQL	4.0.20a	ByteFX.MySqlClient	0.7.6.15073

Database Interface



For information about how to configure a specific database, please refer to the following sections:

- [Using ODBC Databases on page 447](#)
- [Using Microsoft SQL Server on page 448](#)
- [Using ORACLE Databases on page 449](#)
- [Using Microsoft Access Databases on page 451](#)
- [Using SQL Server CE on page 453](#)
- [Using Sybase on page 455](#)
- [Using Microsoft Excel on page 456](#)
- [Using MySQL on page 458](#)

17.2 SQL Relational Databases

SQL relational databases consist of information stored in tables with fields and registers and support SQL commands.

Each database can include one or more tables. Each table is composed of fields (columns) and registers (rows). Typically, the fields are pre-defined and the project adds or reads one or more registers, according to the query condition.

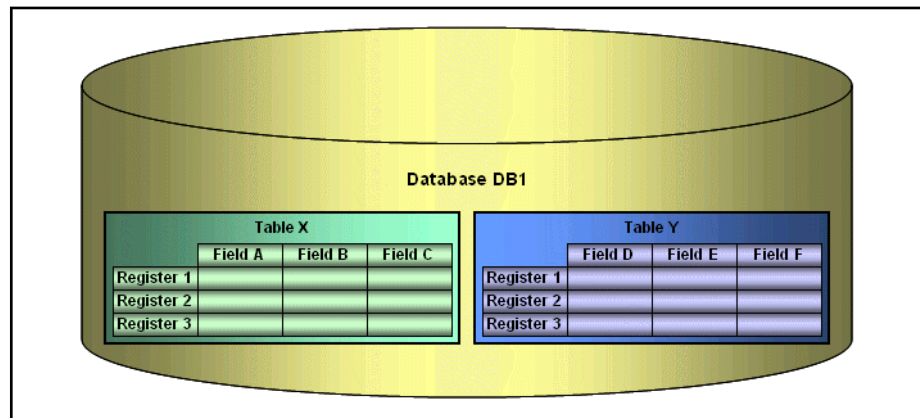


Fig. 17-1: SQL relational databases

"WinStudio" uses database providers (ADO.NET) as interface to SQL relational databases. Database providers are libraries developed to access data from different databases through SQL commands. The ADO.NET provider for a specific database can be provided by the operating system or by the database manufacturer.

The following figure illustrates how "WinStudio" can communicate with different databases using a different database provider for each database.

Database Interface

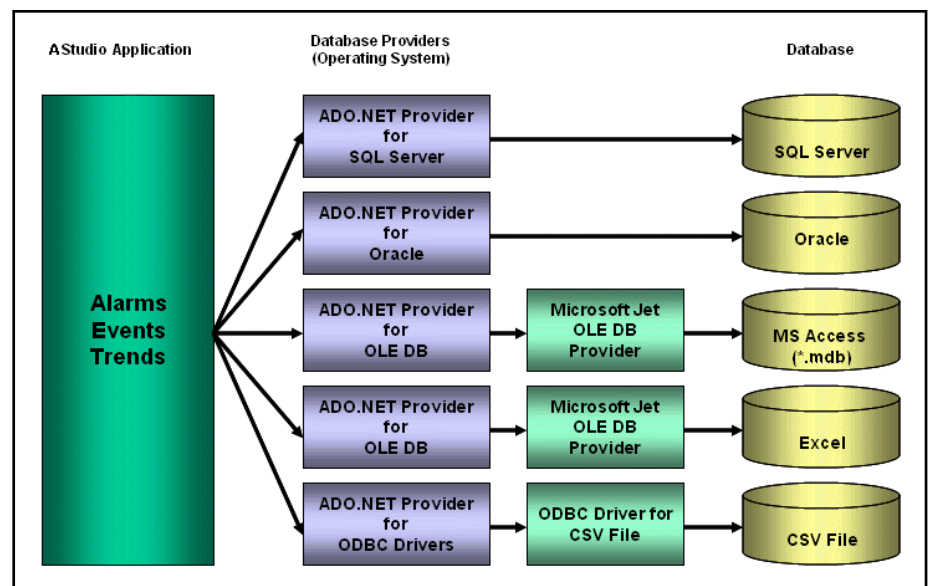


Fig. 17-2: "WinStudio" with different databases...

The previous figure illustrates some of the most popular ADO.NET providers for databases. Notice that the "Microsoft ADO.NET provider for ODBC drivers" provides access to the database through an ODBC driver. Refer to [Database Appendix A: Using ODBC databases on page 447](#). It is also possible that an OLE DB provider is available instead of an ADO.NET provider. The "Microsoft ADO.NET provider for OLE DB" provides access to the database. Using this approach, the Microsoft Jet OLE DB provider provides access to applications in the Microsoft Office package.



It is important to note that "WinStudio" provides the interface for ADO.NET providers. However, ADO.NET providers and/or ODBC drivers/OLE DB providers have to be provided either by the operating system or by the database manufacturer. If the connection string does not refer to a valid ADO.NET provider, the OLE.DB provider is to be used.

Although most projects are typically only linked to one type of database, "WinStudio" provides the flexibility to link each task to another specific database supported. Furthermore, by using this architecture, it does not need to be worried about the specific characteristics of each database (this is mostly handled by the database provider for each database or by the "WinStudio" Database Gateway interface). For this reason, the project settings are mostly the same, regardless of the specific database selected.

17.3 Linking the Database Through a Remote DB Provider

Depending on the architecture of the project, the ADO.NET provider for the SQL relational database may not be available on the same stations where "WinStudio" is running. This scenario is especially common, if the project is run on a Windows Embedded target system (currently, most of the providers are not supported for Windows Embedded). In order to solve this problem, a flexible solution has been developed to configure distributed systems as illustrated in the following figure:

Database Interface

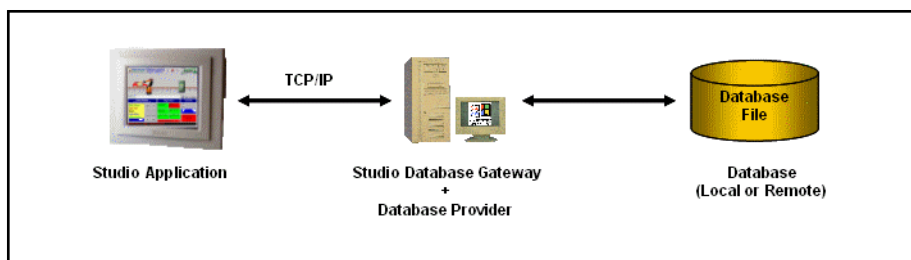


Fig. 17-3: Linking the database through a remote DB provider

The project is running on the server station (where "Rexroth WinStudio" or CEView is installed). The project can communicate with the "WinStudio" Database Gateway (running on a remote computer) via TCP/IP. The gateway implements the interface with the database through the database provider available on the computer where it is running.

The Studio Database Gateway does not require complex configuration. The files **STADOSvr.exe** and **StADOSvr.ini** are just to be copied from the \BIN subfolder of "WinStudio" to any directory of the computer that is to be used as gateway station and that executes the **STADOSvr.exe** program. There are advanced settings associated with the "WinStudio" Database Gateway, however, they are to be changed only under special circumstances. For further information about configuration of the "WinStudio" Database Gateway settings, refer to [Studio Database Gateway on page 432](#).



The Studio Database Gateway is a TCP/IP server for the "WinStudio" project and uses TCP port 3997 by default. If the program **STADOSvr.exe** is executed, a different port number can be specified according to the following syntax:

STADOSvr.exe port number

Example:

StADOSvr 3998

17.4 Studio Database Gateway

The Studio Database Gateway is a TCP/IP server that communicates with databases using the Microsoft .Net framework. It can run either on the same computer that is running the "WinStudio" project or on a different computer. The "Host" entry in the "Database Gateway" pane of the advanced settings (refer to [Database Configuration dialog on page 438](#)) defines whether the gateway is to run on the local computer. For the local computer, either "localhost" or **127.0.0.1** is to be entered as host name. The gateways do not need to be started or stopped as "WinStudio" tasks do this automatically. If the gateway is executed remotely, however, it is to be started manually. To do so, copy the files **STADOSvr.exe** and **StADOSvr.ini** from the \BIN folder to the remote computer and execute **STADOSvr.exe**.

The gateway can be started multiple times for different TCP/IP port numbers. The default port number is 3997 and can be changed by specifying the desired port number in the command prompt (e.g., **StADOSvr 1111**). When running the **StADOSvr**, the following icon is added to the system tray:

Database Interface

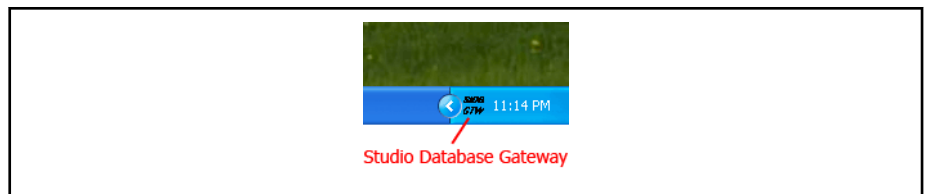


Fig. 17-4: Studio Database Gateway

Right-click on the icon to display the following context menu:



Fig. 17-5: Studio Database Gateway, context menu

The **Hide** option controls whether the "Studio Database Gateway" window is displayed on the desktop. (The gateway software runs continuously after starting it, regardless of whether the window is displayed.) Disable the **Hide** option, to display the window:

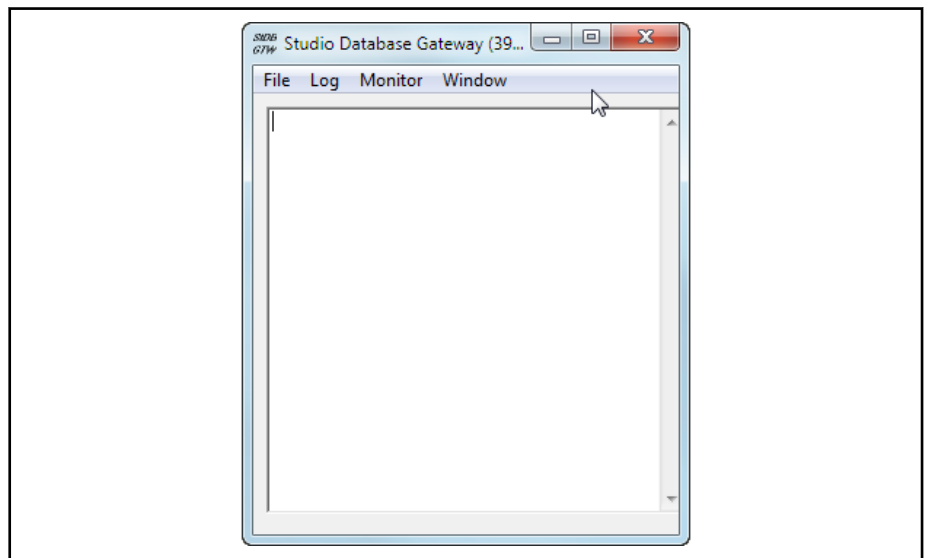


Fig. 17-6: Studio Database Gateway

Any errors that occur during database operation are displayed in this window and also in the "LogWin" window. The messages are reported by exceptions generated by the ADO.NET provider. (For further information on error messages in the gateway module, refer to [Database debugging on page 443](#).)

The output can be configured in this window by using the **Log** menu:

- **Show Log** menu option: Shows the "WinStudio" Database Gateway log files.
- **Options** menu option: Opens the "Configure Messages" dialog.

Database Interface

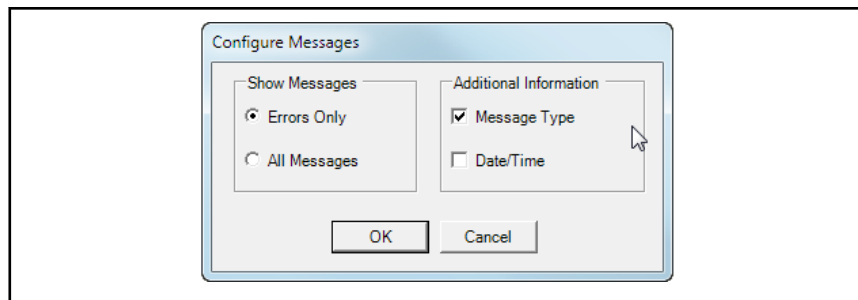


Fig. 17-7: Studio Database Gateway: Configure Messages dialog

- "Show Messages": Select **Errors Only** to show only error messages in the log, or select **All Messages** to show all database messages.
- "Additional Information": In this pane, the display of additional information on each database message can be configured.
 - **Message Type**: Click on (enable) this option to display the type of the message.
 - **Date/Time**: Click on (enable) this option to display the time stamp of the message.

Advanced settings

The Studio Database Gateway offers advanced settings that are configured in the "StADOSvr.ini" file. If there are problems communicating with a specific database, some of these settings probably have to be changed or new providers added to the file. The following parameters are available:

Database Interface

Section of the .INI file	Parameter	Accepted values	Description
Providers (1)	SaveMSec	1 : Disable 2 : Enable 3 : Separate column	This setting specifies the default behavior for the provider when saving milliseconds. The default can be changed in the Advanced Settings in the Database Configuration dialogs.
	Assembly	Any string that contains a .Net framework assembly	Assembly option for all providers. The assembly provides the classes required to communicate with the database. Most providers are inside the System.Data assembly.
	ConnectionClass	Any connection class inside the assembly	The connection class implements the System.Data.IDbConnection interface.
	DataAdapterClass	Any data adapter class inside the assembly	The data adapter class is used on operations where updates to the database are necessary. It has to be compatible with the connection class specified and is to implement IDbDataAdapter.
	CommandBuilder-Class	Any command builder class inside the assembly	The command builder class is also responsible for updates of databases. It has to be compatible with the connection class.
	Provider	Name of the provider	One of the parameters in the connection string is the "provider". The Studio ADO Gateway compares the value on the connection string with the value for this parameter in each provider and defines the proper one to be used.
	ColumnDelimiter-Prefix	Any character or group of characters	Specify a character that is placed before column names in SQL statements
	ColumnDelimiterSuffix	Any character or group of characters	Specify a character to be placed after column names in SQL statements
	TableDelimiterPrefix	Any character or group of characters	Specify a character that is to be placed before table names in SQL statements
	TableDelimiterSuffix	Any character or group of characters	Specify a character that is to be placed after table names in SQL statements
	ValueString	Any string	This value indicates how constants are identified in SQL statements. For Microsoft SQL databases for instance, the value is to be @Value, for ODBC a question mark (?)
	ValueStringPrefix	Any string	This value indicates a prefix to be used before the values. Oracle values, for instance, require this prefix. The SQL statements use value identifiers by using their prefixes. However, the parameters in the connection class do not use the prefix.

Database Interface

Section of the .INI file	Parameter	Accepted values	Description
Providers (2)	ValueAddNumber	0 or 1	Indicates whether a sequential number is to be added to the ValueString to identify the parameter or not. For Microsoft SQL databases, this parameter is to have the value 1, as parameters are identified by using @Value1, @Value2, ..., @ValueN. For ODBC, this parameter is to be 0.
	BoolType	Any string representing a valid data type for the database	When trying to create columns to store Boolean values, the data type specified in this parameter is used. It has to be made sure that the data type specified can save Boolean values.
	IntegerType	Any string representing a valid data type for the database	When trying to create columns to store Integer values, the data type specified in this parameter is used. It has to be made sure that the data type specified can store 32 bit values.
	RealType	Any string representing a valid data type for the database	When trying to create columns to store Real values, the data type specified in this parameter is used. It has to be made sure that the data type specified here can store 64 Real values.
	StringType	Any string representing a valid data type for the database	When trying to create columns to store String values, the data type specified in this parameter is used. It has to be made sure that the data type specified can save the number of characters that are to be saved in the project.
	TimeStampType	Any string representing a valid data type for the database	When trying to create columns to store time stamp values, the data type specified in this parameter is used.
	EnableTop	0 or 1	If this field is set to 1, the ADO places the TOP into the SQL statement to limit the amount of registers required.
	SingleConnection	0 or 1	If this field is set to 1, the ADO opens only one connection to the database, regardless of how many tasks or computers are requesting services from it. Synchronization between the tasks is performed by the gateway and cannot be executed simultaneously if this option is enabled.
Communication	TimeOut	2	Time out to perform insert and update operations
	LongTimeOut	5	Time out to perform connection and query updates
	SyncTimeOut	60	Time out to perform synchronization

Database Interface

Section of the .I NI file	Parameter	Accepted values	Description
Connection	RegBufSize	128	Size of the internal buffer used by the database API.
Options	DisableCloseQuestion	0 or 1	If this field is set to 1, no user confirmation is required to close "StADOSvr.exe". This is important for devices that have alternative methods for exiting and restarting applications.

The parameters are grouped into four sections – "Providers", "Communication", "Connection" and "Options". However, all parameters for database provider configuration are listed in the "Providers" section of the file. The default values are specified in the beginning of the file, using the prefix "Default" in each parameter as illustrated in the following:

Providers

```
[Providers]
DefaultSaveMSec=3
DefaultAssembly=System.Data
DefaultConnectionClass=System.Data.OleDb.OleDbConnection
DefaultDataAdapterClass=System.Data.OleDb.OleDbDataAdapter
DefaultCommandBuilderClass=System.Data.OleDb.OleDbCommandBuilder
DefaultValueString=@Value
DefaultValueAddNumber=1 DefaultBoolType=INTEGER
DefaultIntegerType=INTEGER DefaultRealType=REAL
DefaultStringType=VARCHAR(255) DefaultTimeStampType=DATETIME
DefaultSingleConnection=0
```

The next item on the file lists the number of providers:

Count=5

Providers are identified by the "Provider" parameter followed by a number. When connecting to a database, the Provider parameter in the connection string is compared to the identification of the provider in order to determine which provider is to be used. If there is no provider with the value of the connection string, all the default values are assumed. Besides its identification, each provider can have its own value for each parameter. Again, if no value is specified, the default is used. The following is an example with seven providers:

Example with seven providers

Count=7

```
Provider1=MICROSOFT.JET.OLEDB
SaveMSec1=3
ColumnDelimiterPrefix1=[
ColumnDelimiterSuffix1=]
SingleConnection1=1
```

```
Provider2=SQLOLEDB
ConnectionClass2=System.Data.SqlClient.SqlConnection
DataAdapterClass2=System.Data.SqlClient.SqlDataAdapter
CommandBuilderClass2=System.Data.SqlClient.SqlCommandBuilder
ColumnDelimiterPrefix2=[
ColumnDelimiterSuffix2=]
TableDelimiterPrefix2=[
TableDelimiterSuffix2=]
RealType2=FLOAT
```

```
Provider3=MSDASQL
ConnectionClass3=System.Data.Odbc.OdbcConnection
DataAdapterClass3=System.Data.Odbc.OdbcDataAdapter
CommandBuilderClass3=System.Data.Odbc.OdbcCommandBuilder
ValueString3=?
ValueAddNumber3=0
StringType3=VARCHAR(128)
```

Database Interface

EnableTop3=0

```
Provider4=ORAOLEDB
Assembly4=System.Data.OracleClient
ConnectionClass4=System.Data.OracleClient.OracleConnection
DataAdapterClass4=System.Data.OracleClient.OracleDataAdapter
CommandBuilderClass4=System.Data.OracleClient.OracleCommandBuilder
ValueString4=Value
ValueAddNumber4=1
ValueStringPrefix4=:
BoolType4=Number(1)
IntegerType4=Number(10)
RealType4=Number
StringType4=VARCHAR(255)
TimeStampType4=TIMESTAMP(0)
EnableTop4=0
```

```
Provider5=ASAPROV
Assembly5=iAnywhere.Data.AsaClient
ConnectionClass5=iAnywhere.Data.AsaClient.AsaConnection
DataAdapterClass5=iAnywhere.Data.AsaClient.AsaDataAdapter
CommandBuilderClass5=iAnywhere.Data.AsaClient.AsaCommandBuilder
ValueString5=?
ValueAddNumber5=0
ColumnDelimiterPrefix5=[
ColumnDelimiterSuffix5=]
TableDelimiterPrefix5=[
TableDelimiterSuffix5=]
```

```
Provider6=MYSQLEPROV
Assembly6=ByteFX.MySqlClient
ConnectionClass6=ByteFX.Data.MySqlClient.MySqlConnection
DataAdapterClass6=ByteFX.Data.MySqlClient.MySqlDataAdapter
CommandBuilderClass6=ByteFX.Data.MySqlClient.MySqlCommandBuilder
ValueString6=@Value
ValueAddNumber6=1
StringType6=VARCHAR(128)
EnableTop6=0
```

```
Provider7=MSDAOA
Assembly7=System.Data.OracleClient
ConnectionClass7=System.Data.OracleClient.OracleConnection
DataAdapterClass7=System.Data.OracleClient.OracleDataAdapter
CommandBuilderClass7=System.Data.OracleClient.OracleCommandBuilder ValueString7=Value
ValueAddNumber7=1
ValueStringPrefix7=:
BoolType7=Number(1)
IntegerType7=Number(10)
RealType7=Number
StringType7=VARCHAR(255)
TimeStampType7=TIMESTAMP(0)
EnableTop7=0
```

17.5 Database Configuration

Database connection configuration

The data exchange with an external data base file is carried out analogously to the device drivers. First, configure a new data base connection. Afterward, worksheets can be created in which the created data base connection can be selected and WinStudio tags can be linked to individual data sets of the external database.

To configure a new data base connection, open the context menu of the folder **WinStudio ► Tasks ► External database** and select the entry "Default Database Configuration".



The "External database" folder is only available if "WinStudio 1.5k" or higher is selected as target system in the WinStudio properties and if the device version in the VPx/Vex properties is "13V06" or higher.

Open the configuration dialog for a new database connection using the "Add" button in the overview dialog. Enter a name in the "Name" field that can later be selected in the worksheets.

Database Interface

The "Database Configuration" dialog enables configuration of the necessary settings to link "WinStudio" to an external database file.

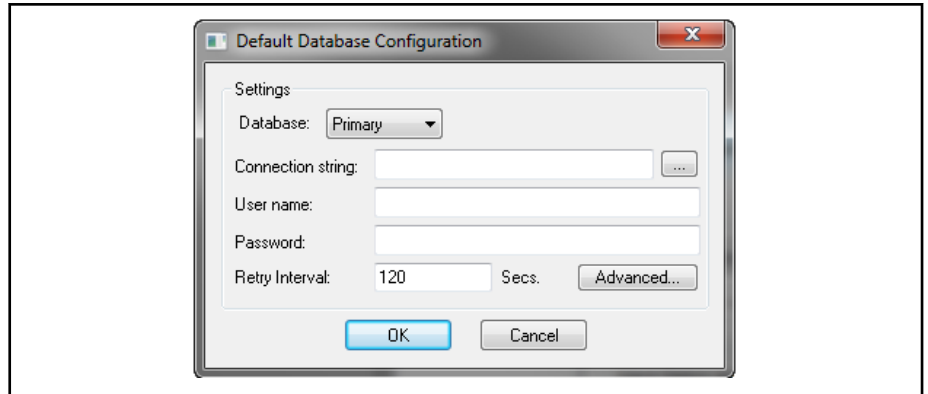


Fig. 17-8: Default Database Configuration dialog

- **Database** combo box: To select either Primary or Secondary. If "Primary" is selected, all settings of the Default Database Configuration apply to the Primary Database interface. Otherwise, they apply to the Secondary Database interface. The Secondary Database can be configured in the following modes:
 - **Disabled:** In this mode, "WinStudio" saves data in the Primary Database only. If the Primary Database is unavailable for any reason, the data is not saved anywhere else. This option may cause loss of data if the Primary Database is not available.
 - **Redundant:** In this mode, "WinStudio" saves data in both Primary and Secondary Databases. If one of these databases is unavailable, "WinStudio" keeps saving data only in the database that is available. If the database that was unavailable becomes available again, "WinStudio" synchronizes both databases automatically.
 - **Store and Forward:** In this mode, "WinStudio" saves data in the Primary Database only. If the Primary Database becomes unavailable, "WinStudio" saves the data in the Secondary Database. If the Primary Database becomes available again, "WinStudio" moves the data from the Secondary Database into the Primary Database.



The Primary and Secondary databases can be different types of databases. However, they have to have the same fields.

Use the Secondary Database to increase the reliability of the system and to use the Secondary Database as a backup when the Primary Database is not available. This architecture is particularly useful when the Primary Database is located in the remote station. In this case, a Secondary Database can be configured in the local station in such a way that data is saved temporarily if the Primary Database is not available (during a network failure, for instance).

- **Use project default:** If this option is checked, "WinStudio" uses the settings in the Default Database for the configured task (Connection string, User name, Password, Retry Interval and Advanced). If this option is disabled, these settings can be configured individually for the current task.
- **Connection string:** This field defines the database where "WinStudio" writes and reads out values as well as the main parameters used when connecting to the database. Instead of entering the connection string

Database Interface

manually, the browse button (...) can be clicked to select the database type from the **Data Link Properties** window.

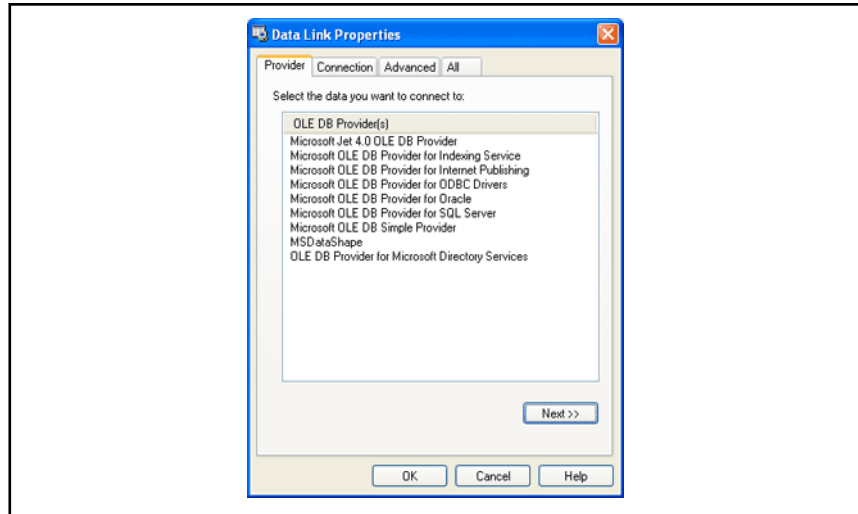


Fig. 17-9: Data Link Properties dialog



The list of database providers shown in the Data Link Properties window depends on the providers actually installed and available on the computer where "WinStudio" is running. Consult the operating system documentation (or the database documentation) for further information regarding the settings of the database provider for the used database.

- **User name:** User name used to connect to the database. The user name entered in this field has to match the user name configured in the database.
- **Password:** Password used to connect to the database. The password configured in this field has to match the password configured in the database.
- **Retry Interval:** If "WinStudio" is unable to connect to the database for any reason, it automatically retries to connect to the database after the seconds configured in this field have passed.
- **Advanced:** Pressing this button provides access to the configuration of advanced settings. For most projects, the default value of these settings does not need to be modified and is to be retained.

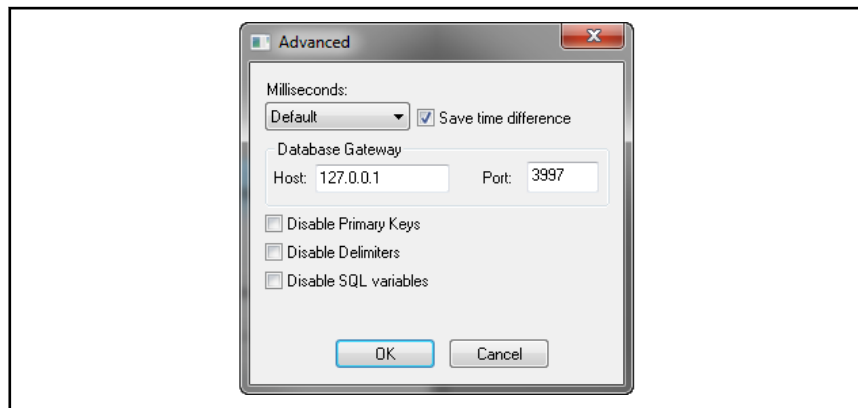


Fig. 17-10: Database configuration: Advanced dialog

– **Milliseconds**

Use this item to set how the milliseconds are saved when saving the date in the database. Different databases save the date in different formats. For example, some databases do not support milliseconds in the **date field**. The following options are available:

- **Default:** Uses the pre-defined format for the current database. The databases that were already tested by "Bosch Rexroth" are preconfigured with the most suitable option. When selecting "Default", "WinStudio" uses the preconfigured setting for the current database type. If a database is used that has not been preconfigured, it is tried to save the milliseconds in a separate field if the Default option is selected.



The default option for each database is pre-configured in the **StADOSvr.ini** file, stored in the **\BIN** sub-directory of "WinStudio". For further information on configuration of the **StADOSvr.ini** file, refer to [Studio Database Gateway on page 432](#).

- **Disable:** Does not save the milliseconds at all when saving the date in the database.
- **Enable:** Saves the milliseconds in the same field where the date is saved.
- **Separate Column:** Saves the milliseconds in a separate column. In this case, the date is saved in one field (without milliseconds) and the number of milliseconds is saved in a different column. This option is displayed if time stamps are to be saved with milliseconds precision and the used database does not support milliseconds for **date fields**.
- **Save time difference:** When this option is activated (default), "WinStudio" saves the time zone of the computer where the project is running in each register of the database. This option has to be enabled to avoid problems with daylight savings time.
- **Database Gateway:** Enter the host name/IP address of the network node where the "WinStudio" Database Gateway is running. The TCP port number can also be specified, however, if the default is not used, the "WinStudio" Database Gateway has to be configured with the same TCP port. For further information on configuration of the advanced settings of the "WinStudio" ADO Gateway, refer to [Studio Database Gateway on page 432](#).
- **Disable Primary Keys:** For some modules, "WinStudio" tries to define a primary key to the table in order to speed up the queries. Enable this option if a database is used that does not support primary keys (e.g. Microsoft Excel).
- **Disable Delimiters:** Select this error handling option to disable the delimiters that are used to format communications with the database. Delimiters can cause problems when a Trend Control or grid creates a query that includes aggregates such as Min and Max.
- **Disable SQL variables:** Select this error handling option to disable SQL variables, such as "@Value1" and "?", that are often used in SQL commands and requests. Some specific database providers do not support these variables.

Table pane

This pane enables configuration of the settings of the table where the data is saved. All tasks can share the same database. However, each task (Alarm,

Database Interface

Events, Trend worksheets) has to be linked to its own table. "WinStudio" does not check this field for invalid configuration. For this reason, ensure that the respective setting is suitable for the used database.

- **Use default name:** If this option is checked (default), "WinStudio" saves and/or loads the data in the table with the default name written to the **Name** field.
- **Create automatically:** If this option is enabled (default), "WinStudio" automatically creates a table with the name displayed in the **Name** field. If this option is disabled, "WinStudio" does not automatically create the table. For this reason, data can only be saved in the database if the database already contains a table with the name that had been preconfigured manually in the field **Name**.
- **Name:** Specifies the name of the table from the database where the history data are saved. To specify a worksheet in a Microsoft Excel spreadsheet file, use the following syntax:
[worksheetname\$]
- **Refresh:** If the set database is currently available, press the **Refresh** button to fill the **Name** combo box with the name of the tables currently available in the database. This way, it is possible to select the table where the history data is to be saved instead of manually entering the table name in the **Name** field.

Runtime pane

In this pane, the runtime values can be defined. The following fields are available:

- **Status (output):** The tag in this field receives one of the following values:

Value	Description
0	Disconnected from the database. The database is not available, its configuration is incorrect or an incorrect operation is executed
1	The database is successfully connected
2	The database is being synchronized

- **Reload (output):** Specify a "reload" tag when using curly brackets in the configuration field. To reconnect to the database using the updated values of the tags, set the tag on this field to 1. "WinStudio" updates the configuration when trying to execute an action in the database, setting the tag back to 0 when finished.

For further information, refer to:

[Configuring a Default Database for All Task Histories on page 442.](#)

17.6 Configuring a Default Database for All Task Histories

A default database can be configured to save the historical data of all tasks in a project. After configuring a default database, either use the default database or configure a new database for this special worksheet when creating a new task worksheet.

Configuration of the connection settings for the default database:

1. Open the properties dialog of the "WinStudio" folder and select the "Options" tab
2. In the "Default Database" pane, click **Configure**. The "Default Database Configuration" dialog is displayed

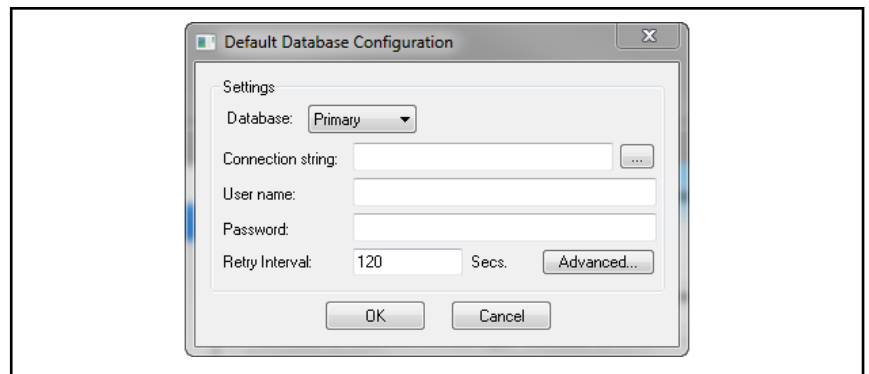


Fig. 17-11: Default Database Configuration dialog

Please refer to [Default Database Configuration dialog on page 438](#) for help regarding the completion the fields in this window.

17.7 Database Debugging

The "WinStudio" database interface provides powerful tools to identify configuration problems in connection with databases. If there are problems in communicating with a database, first enable **Database Messages** in the "LogWin" window. To do so, proceed as follows:

1. Open the WinStudio output window. For this, select the entry **Tools ► LogWin** in the context menu of the "WinStudio" folder.
2. Right-click on the "LogWin" window. By default, it is opened at the bottom of the development environment. In the context menu, click on **Settings**.

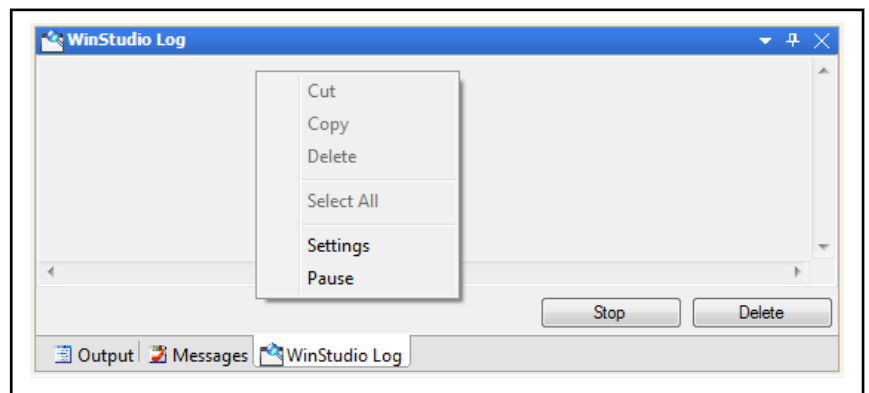


Fig. 17-12: Output window, Settings

3. In the "Log Settings" dialog, enable the **Database Messages** option:

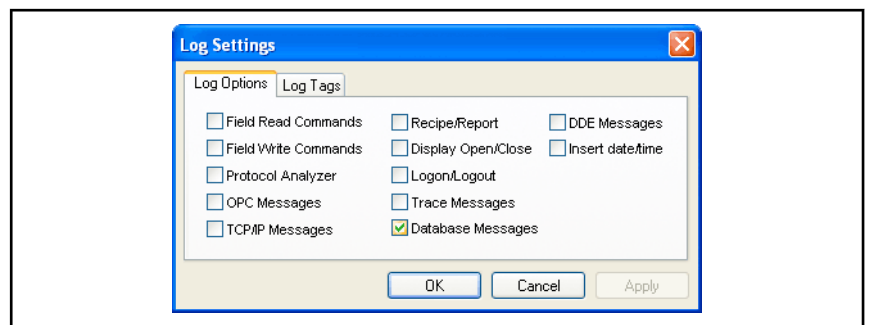


Fig. 17-13: Enabling Database Messages

Database Interface

Frequently asked questions

After enabling this option, the "LogWin" window displays error messages related to the database. The following FAQ section lists some common errors that are displayed in the "LogWin" window.

Q: The database has been configured, however, the runtime modules (alarm, trend and events) are not being saved to the database. In the "LogWin" window, only the following error message is displayed:

Database error: Error adding a new register[CMD_ADD].

A: Most database errors in the "Output" window are followed by additional information such as the SQL command being executed, the connection string and the table name. Error messages such as the one described above usually occur after a more detailed message. For example, if the Trend task fails to add a register to the database because the cable is disconnected, a network error is first received. If the task tries to add more registers before the time specified in the **Retry** field (refer to [Database Configuration dialog on page 438](#)), it is only displayed: **Database error: Error adding a new register[CMD_ADD]**. If the configuration seems to be correct and the cause of the problem is to be limited, reduce the Retry value. In this case, more detailed information is displayed.

Q: On trying to access the MySQL database server, the following message is displayed: Object is not set to an instance of an object.

A: This problem was detected under the following conditions:

- A known bug in MySQL Connector/Net v6.1.2 could not specify the character set correctly; and
- The database table that is accessed does not exist.

To solve this problem, ensure that MySQL Connector/Net v6.2.0 is used and that the accessed table exists in the database.

Q: Why is the database interface automatically closing some connections?

A: By default, the database interface can have a maximum of 1000 connections. If this maximum is exceeded, the oldest connection is automatically closed to allow the new connection and the "LogWin" window displays a detailed message describing which connection was closed and what was the last command executed.

To increase the maximum number of database connections, open the project file (**project_name.app**) in a text editor and change the following setting:

[StDB] MaxConnections="number_of_connections"

Please note that increasing the maximum number of connections may decrease project performance.

Q: The connection string has been configured with the browser and the Data Link Properties window. When clicking the Test button, the message "Test successful" is displayed. However, when I run my project, the database interface displays error messages, and I am not able to save data.

A: The Data Link Properties window uses OLE DB as interface to the database. "WinStudio" Database Interface uses ADO.NET. For this reason, the OLE DB provider can be available on the computer, however, not the ADO.NET provider. It is also possible that an ADO.NET provider is used that is not listed in the StADOSvr.ini file. For further information about adding ADO.NET providers to the "StADOSvr.ini" file, please refer to [Studio Database Gateway on page 432](#).

Q: Why is more than one line updated in the database if data in one line of a grid object is updated?

Database Interface

A: The grid object issues an update command in the database using the values in all the columns for the specific line is to be updated. This problem can occur for lines with double values. If the table has a primary key or any other unique field that is not to be displayed in the "grid" object, it can be added to the **columns**, however, the **width** is to be set to 0. This fixes the problem.

Q: Why do I have to use a separate **column** to save the milliseconds to my database?

A: Some databases do not support milliseconds in the **Time Stamp** field. By default, the "WinStudio" database interface requires an additional column for milliseconds. If the database can process milliseconds or if the milliseconds are not to be recorded, the default behavior can be changed in the advanced settings. Note that some databases are able to store milliseconds, however, with less precision. If different databases with different precisions are mixed in redundant mode, synchronization problems can occur.

Q: The project works fine when running in emulation mode. However, if it is sent to the Windows Embedded device, it cannot communicate with my database.

A: It might be the case that .Net framework or the used provider is not available on the Windows Embedded device. Try to use the gateway remotely by following the instructions in [Linking the database through a remote DB provider on page 431](#).

Q: Why is the message "Error to create connection class" received when trying to connect to the database?

A: The .Net provider that is to be used is not installed on the computer. This error message is usually followed by the provider name. For example, if the Sybase database is used, the message is followed by "[iAnywhere.Data.AsaClient.AsaConnection]". The provider is the iAnywhere.Data.AsaClient. Check if the provider is installed under "**Control PanelAdministrative ToolsMicrosoft .Net framework x.x configuration**". The provider is generally listed in the "Assembly Cache".

Q: What if the provider assembly is available (usually a .dll file), however, it is not listed in the "Assembly Cache"?

A: If the assembly does not have a strong name, it can be registered in the "Assembly Cache" by means of the "gcautil" program. This problem is also generally solved by copying the assembly to the same folder as the "StA-DOSvr.exe" (typically [...]\WinStudio\Bin).

Q: Under a specific condition, the table cannot be accessed from the grid. However, if no condition is applied, it works fine. What is the problem?

A: The following items are to be checked:

1. Follow the [Debugging steps on page 443](#), and check the log for error messages. An error message can indicate that an error has been made. For example, input with an incorrect column name or specifying an invalid data format.
2. In some databases, problems occur if reserved words are used as column names. Therefore, column names such as Time, Date, Numeric, etc. are to be avoided
3. If the column name starts with AND or OR (e.g., ORange), enter the name surrounded by square brackets.

For example:

[ORange]=10 instead of ORange=10.

Database Interface

4. If SQL Server CE is used, problems may occur when querying String fields. It has been identified that filters do not work with NCHAR data types. However, they work if these fields are declared as **NVARCHAR(<Number>)**.

Try to recreate the table using this data type.

Example command creating a table with strings for querying:

```
CREATE TABLE Table1 (Name NVARCHAR(128), Age Numeric, Sex NVARCHAR(1))
```

ORACLE Q: If the connection to an ORACLE database is lost, it does not recover. The following message is received in LogWin:

"Database: Error: ORA-03114: not connected to ORACLE".

Is this a problem with the "WinStudio" database interface?

A: The Oracle .Net provider has a problem managing the connection pool. A QFE 830173 has to be installed. At the time this document was created, further information about this problem could be found [here](#) (external link).

Q: When trying to access the database, the following error message is displayed:

ORA-00162: external dbid length 19 is greater than maximum (16).

What is to be done?

A: At the time this document was created, there was a problem with the Oracle .NET provider. the **Server Name** (SERVER/TNS) could not exceed 16 characters. To fix this problem, try to reduce the **Server Name** field.

One way of doing this is to edit the file `\\WINDOWS\\system32\\drivers\\etc\\hosts` by adding an entry with a shorter server name.

For example, the server name specified by **192.168.89.98** has 13 characters. By adding the following line in the file, it could be reduced to 3:

192.168.89.98 ora

Respectively, the **Server Name** could now be configured using ORA/TNS instead of **192.168.89.98/TNS**.

MYSQL Q: Accessing the database from a local computer does not cause any problems. However, if the project is moved to a remote device, the message **Access Denied** is received.

A: Each user in a MySQL database has a property associated with it that indicates the computer from which it can get access to the database. By default, this property is set to "localhost", i.e., the database can only be accessed from the local computer. For information about changing this setting, refer to the MySQL manual.

Q: Sometimes when trying to synchronize a remote MySQL database with a local MySQL database or to use application redundancy, a connection to the ADO.NET interface is opened and never closed.

A: Open the [Database Configuration dialog on page 438](#) and disable the **Automatically Create** option.

SYBASE Q: The Sybase database has been configured with the Browse button. When clicking on the Test button, the test is successful. However, if the project is to be executed, the following error is displayed:

Database: Error: Parsing error: DSN 'MyDatabase' does not exist.

What is the problem?

A: Please refer to [Database Appendix F – Using Sybase on page 455](#) for more information about this problem.

Database Interface

Q: Why is the error **Error to create connection class [iAnywhere.Data.Asaclient.AsaConnection]** displayed on trying to connect to the Sybase database?

A: The **ADO.NET** provider is not installed on the computer. The database setup program has an option to install the provider. Execute the setup program again and ensure that this option is enabled.

SQL SERVER CE

Q: Why does the gateway display **TypeLoad error** when trying to access the SQL Server CE database?

A: This problem usually occurs if the SQL Server CE .NET provider is not installed on the CE device.

Q: What is the meaning of the error message **There is a file sharing violation. A different process might be using the file?**

A: Another program with the SQL Server CE database is running. For example, this error occurs if the SQL Server CE configuration software is used.

17.8 Appendices

17.8.1 Using ODBC Databases

Almost every database provides an ODBC interface that can be used to communicate with it. The database functions provided by "WinStudio" can be used with ODBC drivers via the ADO.NET interface for ODBC. In order to use this option, Microsoft .NET framework 1.1 or higher has to be used.

Microsoft .NET framework 2.0 is automatically installed with "Rexroth WinStudio v7.2" or higher.

The [Database Configuration dialog on page 438](#) enables to provide connection strings that connect to an ODBC DSN. The connection string can be created automatically by clicking on the Browse button (...). If the "Data Link" window is displayed, the option Microsoft OLE DB Provider for ODBC Drivers is to be selected:

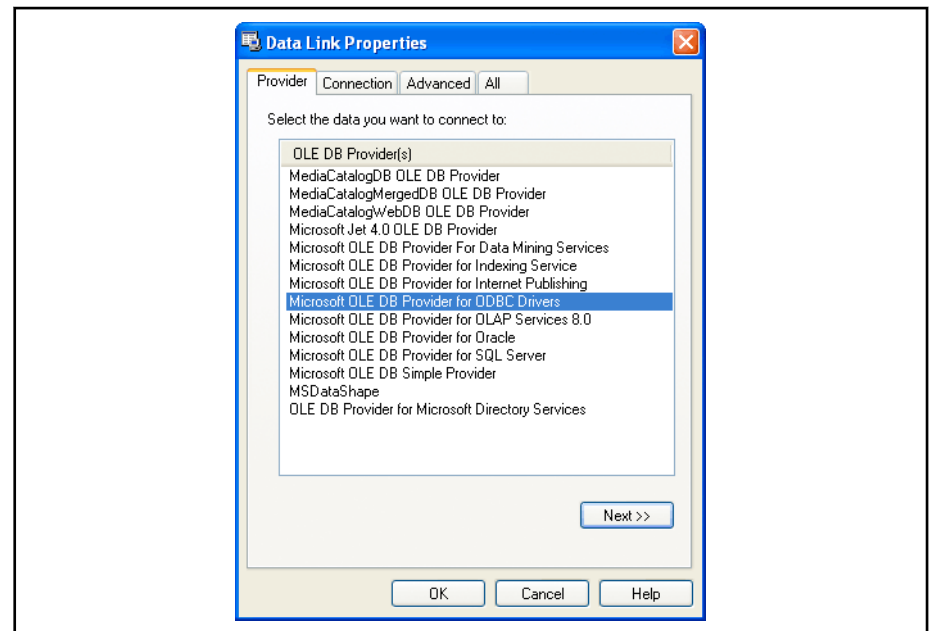


Fig. 17-14: Data Link Properties, Provider – ODBC

Click on **Next** to open the following window:

Database Interface

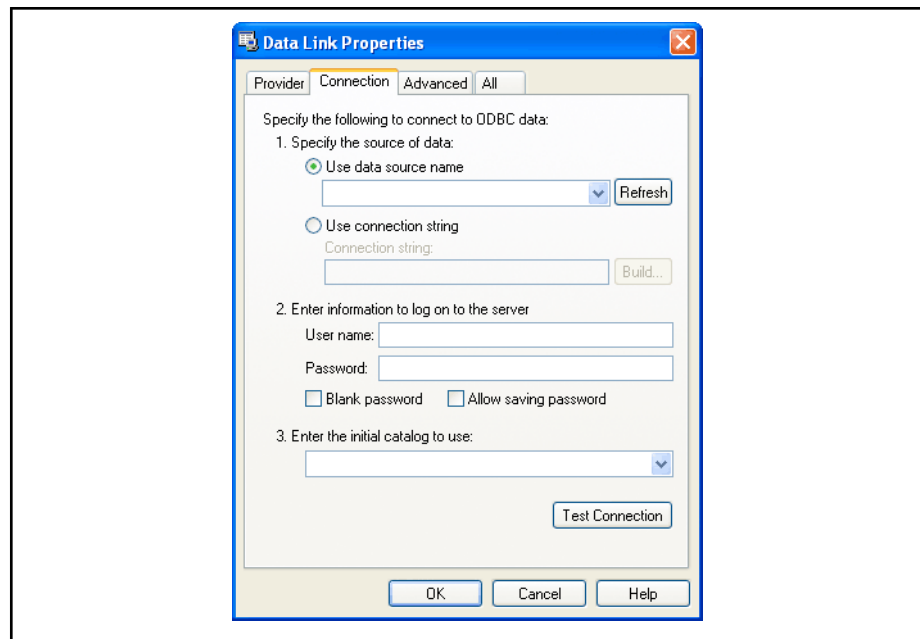


Fig.17-15: Data Link Properties, Provider – Connection

Select the DSN to connect to and click on **OK**. To specify the user name and password on this window instead of specifying it on the "Object Properties" dialog, the **Allow saving password** checkbox has to be enabled.

17.8.2 Using Microsoft SQL Server

"WinStudio" database interface enables retrieval and storage of information on Microsoft SQL Server databases. To configure the SQL Server database, proceed as follows:

1. Click on the **Browse** button in the [Database Configuration dialog on page 438](#). The following window is displayed:

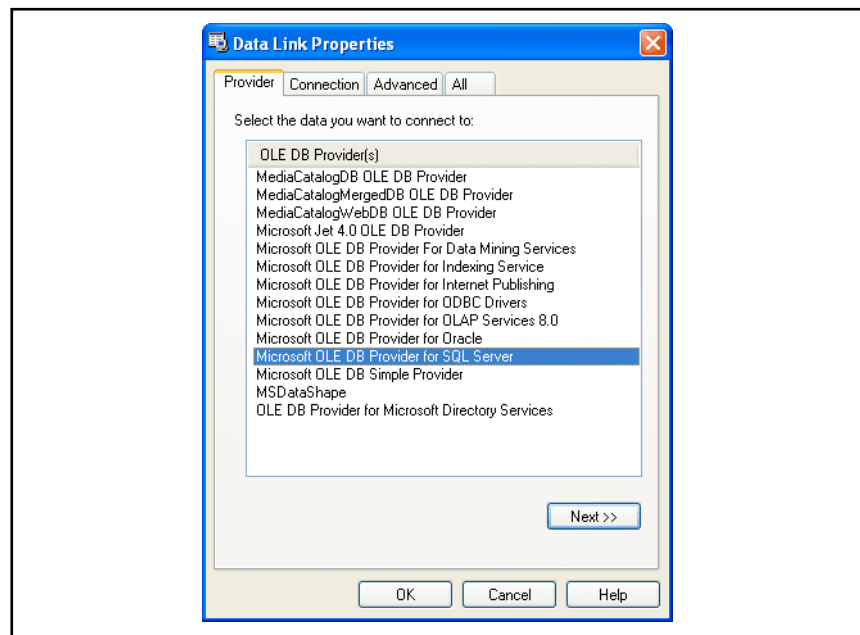


Fig.17-16: Data Link Properties, Provider – SQL Server

2. Select the Microsoft OLE Provider for SQL Server and click on **Next**. The following window is displayed:

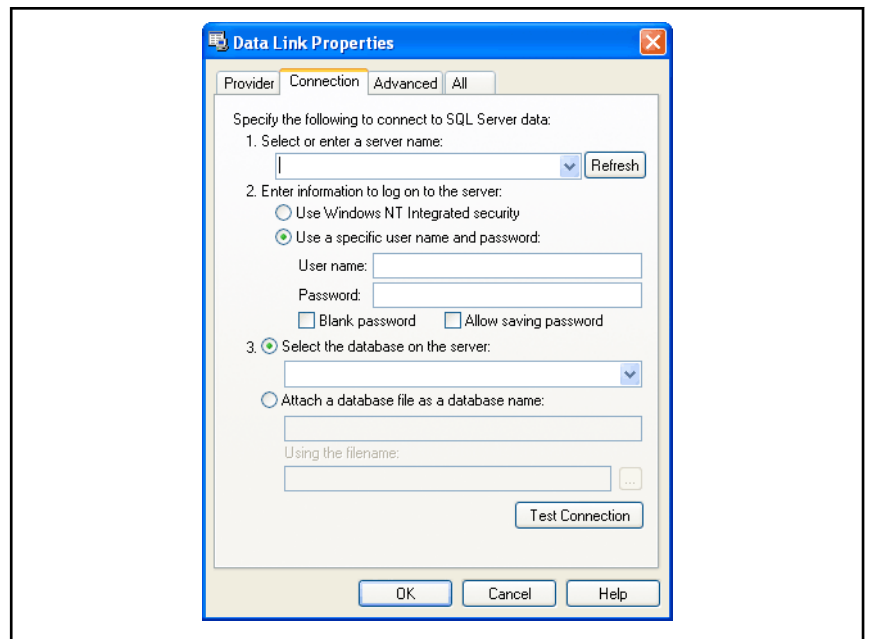


Fig. 17-17: Data Link Properties, Connection – SQL Server

3. Enter the database information into the fields of this window. If Windows NT Integrated Security is not to be used, make sure to enable the **Allow saving password** checkbox when closing the "Data Link Properties" window.

4. Click on **OK** to finish the "Connection Strings" configuration.

The connection string should be very similar to this one:

Provider=SQLOLEDB.1; Integrated Security=SSPI; Initial Catalog=MyDatabase;

Data Source=192.168.23.200



These procedures have been tested using Microsoft SQL Server 2000.

17.8.3 Using ORACLE Databases

The "WinStudio" database interface enables retrieval and storage of information on ORACLE relational databases.

To configure the ORACLE database, proceed as follows:

1. Click on the **Browse** button in the [Database Configuration dialog on page 438](#). The following window is displayed:

Database Interface

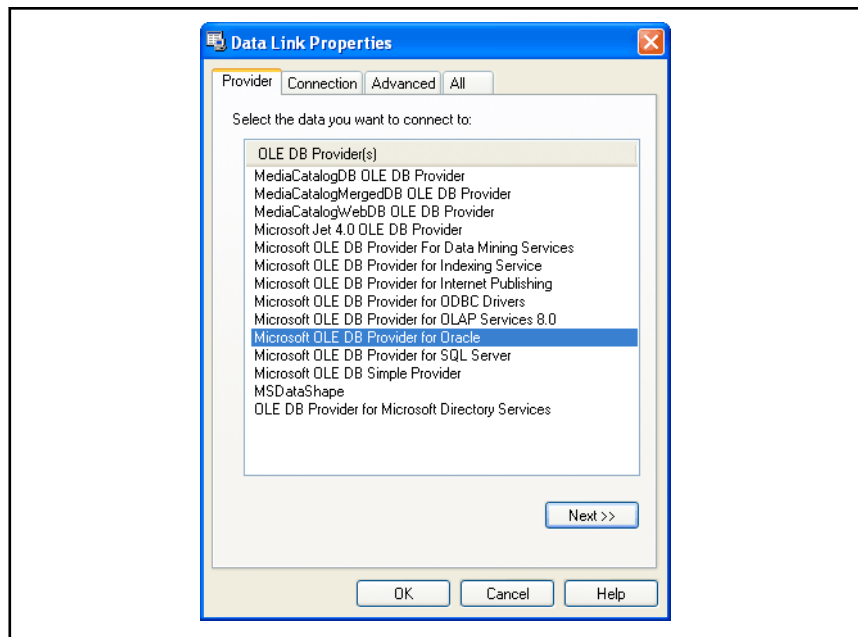


Fig. 17-18: Data Link Properties, Provider – Oracle

2. Select the Microsoft OLE Provider for Oracle and click on **Next**. The following window is displayed:

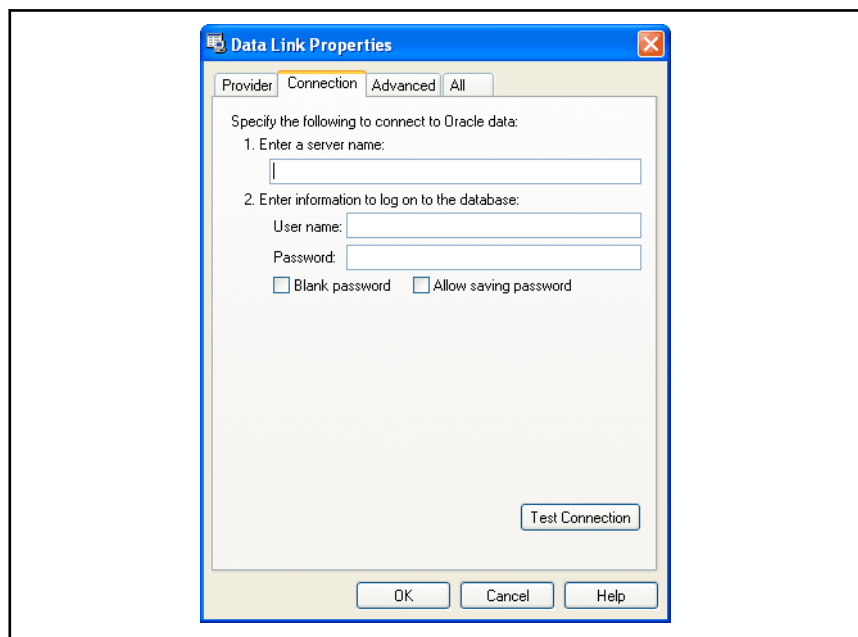


Fig. 17-19: Data Link Properties, Connection – Oracle

3. Enter the database information into the fields of this window. Remember to check the **Allow saving password** checkbox to save the password when the "Data Link Properties" window is closed. The server name information has the following format:

Server / TNS

Where:

- **Server:** Computer where the Oracle database is running
- **TNS:** Oracle TNS name

NOTICE

At the time that this document was created, the **Server Name** field could not be configured with more than 16 characters.

If more than 16 letters are specified, the following error occurs:

ORA-00162: external dbid length 19 is greater than maximum (16).

Whereas 19 refers to the number of characters in the **Server name**.

For more useful information on avoiding this problem, refer to [Database debugging on page 443](#).

4. Click on **OK** to finish the "Connection Strings" configuration.



These procedures have been tested using ORACLE 10g Release 1.

17.8.4 Using Microsoft Access Databases

The "WinStudio" database interface enables storage and retrieval of information in Microsoft Access database (ACCDB) files.

Access has to be installed on the same computer as "WinStudio". "WinStudio" can directly read from or write to existing ACCDB files. However, Access has to be used to create the initial file – either a blank database for new data or a populated database for reference data. After creating the file, it can be moved to any location and a connection to it can be established.



This procedure has been last tested using Microsoft Access 2007 (12.0.6211.1000).

To establish a connection between the "WinStudio" project and an ACCDB file, proceed as follows:

1. In the [Database Configuration dialog on page 438](#), click on the Browse button (...). The "Data Link Properties" are displayed:

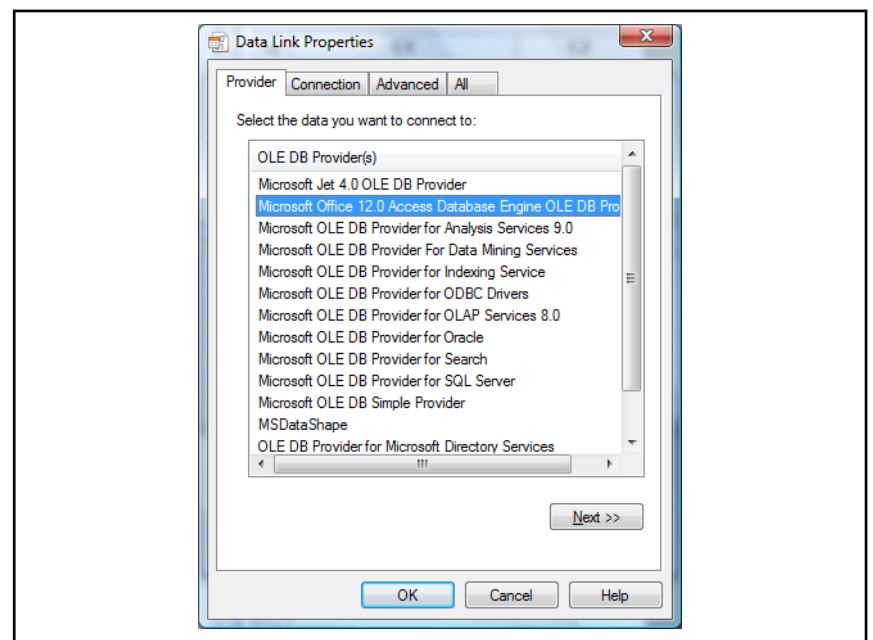


Fig. 17-20: Selecting the OLE DB provider

Database Interface

2. Select the respective OLE DB provider for the database:
 - For a Microsoft Access 2003 database file, select **Microsoft Jet 4.0**.
 - For a Microsoft Access 2007 database file, select **Microsoft Office 12.0 Access Database Engine**.
3. Click on **Next**.

The "Connection" tab of the dialog is displayed:

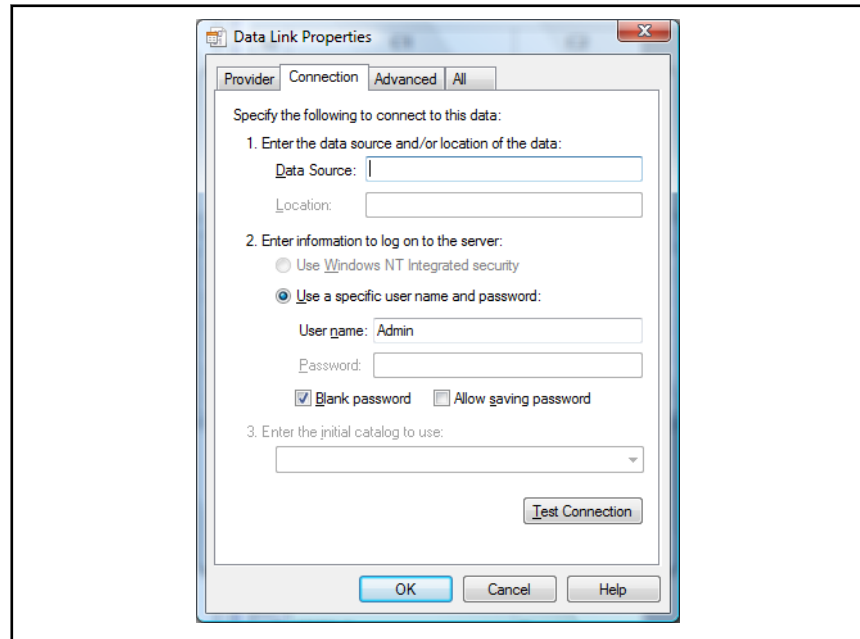


Fig. 17-21: Specifying the location of the file

4. In the **Data Source** text field, enter the complete file path for the ACCDB file. Although the file can be located anywhere on the computer or network, it is recommended to keep it in the project folder. For example:
C:\Users\username\My Documents\WinStudio-Projects\project_name\Database1.accdb
 In this case, however, the file path has to be updated if the project folder is moved.
5. If required, specify the **User name** and **Password** for the ACCDB file.
6. Click on **Test Connection**.
 If a connection can be successfully established, a respective message is displayed.
7. Click on **OK** to save the changes and return to the "Database Configuration" dialog.



Ensure that the **Disable Primary Keys** option in the "Database Configuration" dialog is enabled (checked). If not, "WinStudio" cannot connect to the Access database



Desktop Office applications such as Microsoft Access and Microsoft Excel cannot efficiently handle large amounts of data. Trying to save all historical project data in an Access database or Excel spreadsheet, results in slow queries that may lead to unexpected results. For this reason, it is recommended to use Access or Excel only as secondary database with the **Store and Forward** option enabled or to manage data with third-party software.

To handle large amounts of historical data, it is recommended to use either the proprietary format of "WinStudio" or a dedicated relational database such as Microsoft SQL Server or ORACLE.

17.8.5 Using SQL Server CE

Using SQL Server CE, General Information

"WinStudio" projects running in CEView can communicate with Microsoft SQL Server CE (SQL CE) databases on the same device by using the OLE DB Provider for SQL Server CE. This provider has to be installed on the Windows Embedded device.

As the provider is only installed on the device and not on the development workstation, it cannot be selected in the [Data Link Properties dialog on page 438](#) when developing the project. Instead, the connection string has to be entered manually using the following format:

Version	Connection string
SQL CE 2.0	Provider=SQLCE; Data Source=database_path
SQL CE 3.0	Provider=SQL_CE3.0; Data Source=database_path
SQL CE 3.5	Provider=SQL_CE3.5; Data Source=database_path



The database interface has been fully tested with these versions of SQL CE. However, other versions might also be supported. Please contact the customer support for more information.

Examples:

- Accessing a data base file at a fixed storage location, e.g. **\Harddisk\MyDatabase.sdf**:

Provider=SQLCE; Data Source=\Harddisk\MyDatabase.sdf

The exact path depends on the organization of the non-volatile memory on the device. Please refer to the documentation by the manufacturer.

- Accessing a database file with a storage location defined by the value of the "DatabaseFile" String tag:

Provider=SQLCE; Data Source={DatabaseFile}

The curly brackets ("{}") indicate that it is a tag reference.

Please note that SQL CE is only intended for lightweight databases and simple transactions. It cannot efficiently handle large amounts of data, such as the alarm and event histories. In this case, it is recommended to either use the proprietary history format or a more powerful relational database like the full version of Microsoft SQL Server. If SQL CE "has to be" used, it is recommended to use it only as a secondary database with the **Store and Forward** option enabled. For more information, refer to [Saving the alarm history/event log to an external database on page 254](#).

Database Interface

NOTICE

When using a [database worksheet on page 328](#) or the [DB/ERP functions on page 471](#) to access a SQL CE database, please note that...

- Column/field names have to fit exactly or the database commands fail; and
- [WinStudio data types on page 113](#) are converted to potentially unexpected SQL data types.

The following table describes the conversion:

WinStudio	SQL CE
 Boolean	int
 Integer	int
 Real	real
 String	nvarchar

Installing SQL Server on a Windows Embedded Device

Microsoft SQL Server is not included by the manufacturers of most Windows Embedded and Windows Mobile devices. If the "WinStudio" project requires SQL Server, it may have to be downloaded and installed on the device.

Before proceeding, consult the documentation provided by the manufacturer to check whether the device includes Microsoft SQL Server. If this is the case, the software does not need to be installed again unless it is causing problems.

If not, check at least if the device meets the [System Requirements on page 37](#) for running a "WinStudio" project and note the device processor.

As only the device manufacturer can "permanently" install software on the device with Platform Builder, the CAB files are actually only copied to the non-volatile memory of the device and then configured to load every time it starts.



When this documentation was created, the latest stable version of SQL Server was **Microsoft SQL Server Compact Edition 3.5 Service Pack 2**.

1. Download Microsoft SQL Server Compact Edition to the desktop from the following website: <http://www.microsoft.com/sqlserver/en/us/editions/compact.aspx>
2. Extract and install the software from the file on the desktop. A new directory is created with the Microsoft SQL Server files including customized files for various device processors.
3. Open the directory: **C:\Program Files\Microsoft SQL Server Compact Edition\v3.5\Devices\wce500**
4. In the directory, select the correct sub-directory for the respective device processor.
For example, if the device uses a x86 processor, select the "x86" sub-directory.
5. In the sub-directory, select the CAB files to be copied to the device. For example, for x86, select the following files:

- sqlce.dev.ENU.wce5.x86.CAB
 - sqlce.repl.wce5.x86.CAB
 - sqlce.wce5.x86.CAB
6. Copy the CAB files into the non-volatile memory of the device and note the storage location. The exact procedure to copy files to the device depends on the respective installation. Potentially, desktop synchronization, a network connection or an USB stick may be used. Check the documentation of the manufacturer for information on the available options.
 7. On the desktop, create a new text file named **install.bat**.
 8. Edit **install.bat** to create a batch script that loads the CAB files on start-up.
Continuing with the x86 example, edit the file to include the following code:
wceload /noui /delete 0 "filepath\sqlce.dev.ENU.wce5.x86.CAB"
wceload /noui /delete 0 "filepath\sqlce.repl.wce5.x86.CAB"
wceload /noui /delete 0 "filepath\sqlce.wce5.x86.CAB"
...where **filepath** is the location to which the CAB files were copied.
For further information, refer to [the MSDN article about the Wceload tool](#).
 9. Copy **install.bat** to the **\Windows\Startup** directory. If the **\Windows\Startup** directory is not available on the device, follow the instructions in the documentation of the manufacturer for configuration of startup elements.
 10. Restart the device. After selecting and installing the files properly, SQL Server runs on startup and can be accessed with the "WinStudio" project.

17.8.6 Using Sybase

The AsaClient provider has to be installed on the computer. The tests with "WinStudio" have been performed using the architecture described in section [Linking the database through a remote DB provider on page 431](#).

If the "Browse" button is used to automatically generate the connection string, the string returned has the following format:

Provider=ASAProv.90; Data Source=Test

For this format, an ODBC DSN with the same name as the **Data Source** (in this case, "Test") has to be created to communicate with the database. If the DSN is not created, the following error is displayed in LogWin when connecting to the database:

Database: Error: Parsing error: DSN 'Test' does not exist.

To avoid an ODBC DSN error, the connection string can be manually used for access as illustrated in the following example:

Provider=ASAProv.90; DBF=C:\ Test.db

Database Interface



These procedures have been tested using Sybase Server Anywhere 9.0.1.1751.

17.8.7 Using Microsoft Excel

The "WinStudio" database interface enables storage of information in and retrieval of information from Microsoft Excel spreadsheet (XLS or XLSX) files.

Excel has to be installed on the same computer as "WinStudio". "WinStudio" can directly read from or write to existing XLS/XLSX files. However, Excel has to be used to create the initial file – either a blank spreadsheet for new data or a populated spreadsheet for reference data. After creating the file, it can be moved to any location and a connection to it can be established.



This procedure has been last tested using Microsoft Excel 2007 (12.0.6331.5000).

To establish a connection between the "WinStudio" project and the XLS/XLSX file, proceed as follows:

1. In the [Database Configuration dialog on page 438](#), click on the Browse button (...). The "Data Link Properties" are displayed:

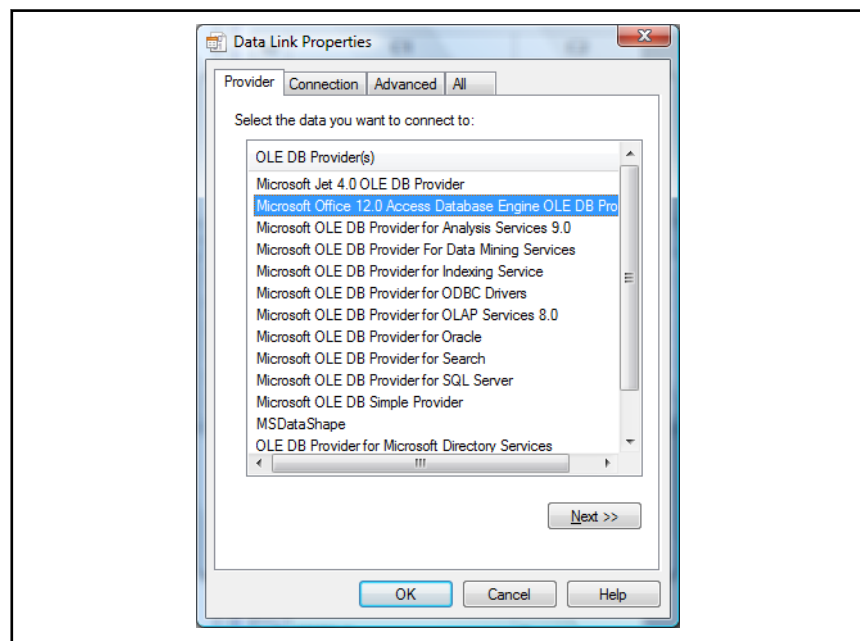


Fig. 17-22: Selecting the OLE DB provider

2. Select the respective OLE DB provider for the database:
 - For a Microsoft Excel 2003 spreadsheet file, select **Microsoft Jet 4.0**.
 - For a Microsoft Excel 2007 spreadsheet file, select **Microsoft Office 12.0 Access Database Engine**.
3. Click on **Next**.

The "Connection" tab of the dialog is displayed:

Database Interface

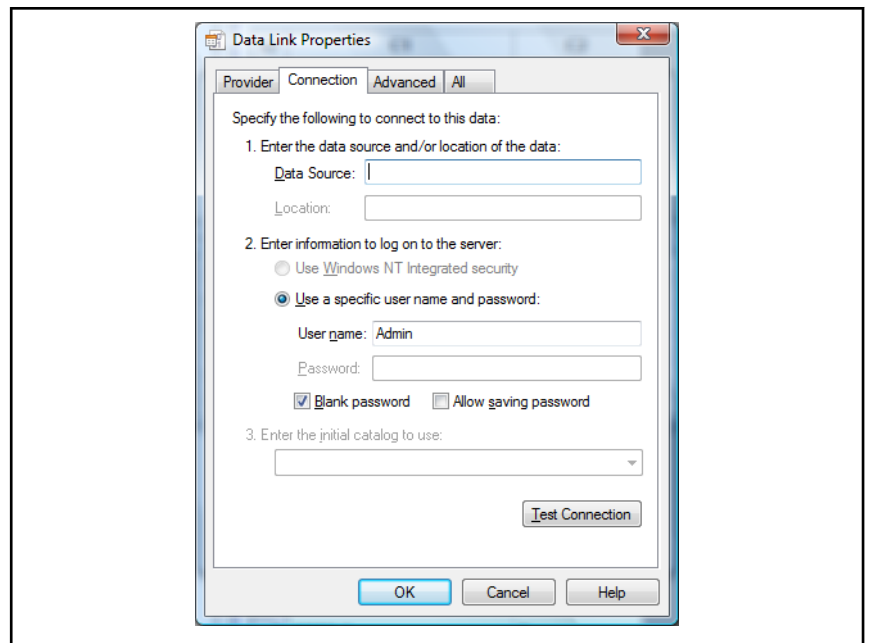


Fig. 17-23: Specifying the location of the file

4. In the **Data Source** text field, enter the complete file path of the XLS/XLSX file. Although the file can be located anywhere on the computer or network, it is recommended to keep it in the project folder. For example:
C:\Users\username\My Documents\WinStudio-Projects\project_name\workbook1.xls
In this case, however, the file path has to be updated if the project folder is moved.
5. Click on the **All** tab.
6. Select **Extended Properties** and click on **Edit Value**. The "Edit Property Value" dialog is displayed.
7. In the **Property Value** text field, enter one of the following values:
 - For a Microsoft Excel 2003 spreadsheet file, enter "Excel 11.0".
 - For a Microsoft Excel 2007 spreadsheet file, enter "Excel 12.0".
8. Click on **OK** to close the "Edit Property Value" dialog.

Database Interface

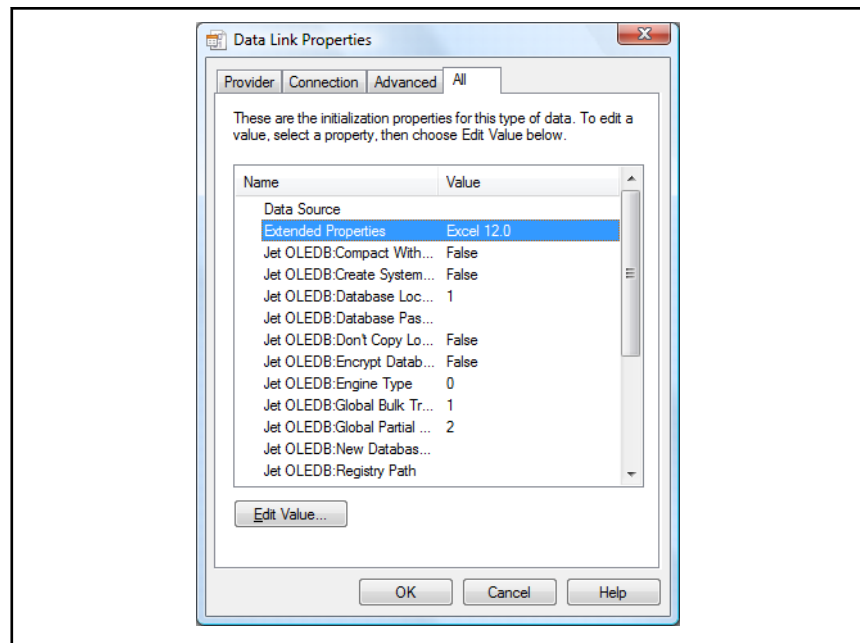


Fig. 17-24: Editing the value of extended properties

9. Click on the **Connection** tab.
10. Click on **Test Connection**. If a connection can be successfully established, a respective message is displayed.
11. Click on **OK** to save the changes and return to the "Database Configuration" dialog.



Ensure that the **Disable Primary Keys** option in the "Database Configuration" dialog is enabled (checked). If not, "WinStudio" cannot connect to the Excel spreadsheet.



Desktop Office applications such as Microsoft Access and Microsoft Excel cannot efficiently handle large amounts of data. Trying to save all historical project data in an Access database or Excel spreadsheet, results in slow queries that may lead to unexpected results. For this reason, it is recommended to use Access or Excel only as a secondary database with the **Store and Forward** option enabled or to forward data to third-party software.

To handle large amounts of historical data, it is recommended to use either the proprietary format of "WinStudio" or a dedicated relational database such as Microsoft SQL Server or ORACLE.

17.8.8 Using MySQL

"WinStudio" can communicate with MySQL databases, however, an ADO.Net provider for MySQL has to be installed.

The provider required by "WinStudio" is MySQL Connector/Net. When this document was created, the necessary software could be downloaded from [the official MySQL site](#). (Please note that the linked site is a third-party site and may change without notice.)

After the provider is installed, the [Database Configuration dialog on page 438](#) can be used to configure a MySQL database connection. However, unlike for other database types, the "Data Link Properties" dialog (which is accessed by clicking ... to the right of the **Connection string** box) cannot be used to create

Database Interface

the connection string. Instead, the connection string has to be entered directly using the following format:

Provider=MYSQLCLIENT; Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword;

The following optional parameters can be added to the connection string:

Optional parameters for the MySQL Connector/Net connection string

Parameter	Description
Port=number;	Specifies the port to be used for connection. The default port is 3306. However, any port can be specified as long as it matches the server configuration. If a port of -1 is specified, the connection uses the Named Pipes network protocol (see "Protocol" below).
Server=myServerAddress1 & myServerAddress2 & ... & myServerAddressN;	Uses any server in a replicated server configuration.
Encryption=true;	Enables SSL encryption for all data sent between the client and the server. The server requires a valid certificate installed.
Encrypt=true;	An alternative to "Encryption" above, in case of errors.
Default Command Timeout=milliseconds;	Specifies a default command timeout for the connection. This does not supercede any timeout properties on individual commands.
Connection Timeout=seconds;	Specifies how long the client waits for a server connection before terminating the attempt.
Ignore Prepare=true;	Instructs the database provider to ignore Command.Prepare statements to prevent corruption by server commands.
Protocol=myProtocol;	Specifies the network protocol to be used. The default is "socket" or "tcp". However, "pipe" can be specified for a Named Pipes connection or "memory" for a shared memory object.
Shared Memory Name=MySQL;	Specifies the name of the shared memory object to be used for communication. (This parameter applies only if the "Protocol" parameter above is set to "memory".)
CharSet=UTF8;	Specifies the character set to be used to encode queries to the server. Please note that query results are encoded in the same character set used to record the data itself.



These procedures have been tested using MySQL v5.1.11 and MySQL Connector/Net v6.2.0.

18 Troubleshooting and FAQs

18.1 General Troubleshooting

If technical assistance is required, there is certain information required before contacting Technical Support. Regardless of the problem, the sequence of events that led to the detection of the problem is useful. It has to be explained as detailed as possible and is not to be improvised as it may strongly influence the time for and process of troubleshooting.

Before contacting Technical Support

Some things to try before contacting Technical Support:

- **Consult the documentation** The help function is available in all WinStudio editors. Press "F1" to open the WinStudio help.
- **Consider recent changes on the system** If something used to work, think about what may have changed. New software installation or general system changes can affect performance and general functionality of other software on the system.
- **Try reproducing the problem in a new file** If the problem can not be reproduced in a new test file, compare the new file with the original file to find and eliminate any differences. This helps to narrow down the cause of the issue.
- **Try reproducing the problem on another computer** If the problem does not occur on another computer, compare the two systems. If this is the case, there is most likely a system conflict.

Verifying the project

If any tags are changed, reordered or deleted in the tags database or if any settings in the Web tab of the Project Settings dialogue are reconfigured, the project has to be verified to update all of the screens and worksheets to the current state of the database. In the WinStudio context menu, select **Verify Application**.

Related publications

The **WinStudio Getting Started Guide** is designed for first-time users. This publication contains information about the basic functions of "Rexroth WinStudio" and it is provided in the **Documentation** folder on the "Rexroth WinStudio" installation CD.

The driver "User Guides" explain how to configure the direct communication drivers according to their unique protocol characteristics. A documentation is provided for each driver. The documentation of a driver can be displayed by selecting the respective driver folder in the "Communication" folder and pressing "F1". Alternatively, the driver description can be found in the [...]\WinStudio\Drv folder.

Contacting Technical Support

If no answer to the technical question can be found in the product documentation or help system, the Bosch Rexroth Technical Support specialists are available to assist any customer at current product maintenance.

Please try to define the problem before contacting Technical Support to be able to repeat the steps that led to the problem and specifically identify when and how the problem occurred. The support representative needs to know exactly what the problem is in order to provide help. These steps help to pinpoint and solve the problem more quickly.

Please have the following information available:

- Hardware environment: available memory, processor type, output device
- Software environment: operating system, version of Windows®, network platform

Troubleshooting and FAQs

- Product name, version number and product serial number
- Memory space installed on the system
- Free hard disk space on the system
- Screen resolution (screen size in pixels, for example, 1024 x 768)
- Screen color depth (number of colors or bits, for example, 256 colors or 8-bit color)
- Graphics card manufacturer, model name and driver version number
- Sound card manufacturer and model name
- A list of external devices connected to the computer
- Brief description of the problem or error and the specific text of any error messages
- Description of the steps taken to troubleshoot the issue. For example, how many computers have been tested and whether the issue is reproducible in a new file
- Steps to reproduce the issue if it is reproducible. If the issue is not reproducible, it may be a development issue rather than an issue with the product.

If the project crashes completely during runtime, a debugging report is generated under:

`[...]\project_name\Web\Dump\Windump.dmp.`

Please have this file ready to send to Technical Support for analysis.

When contacting Bosch Rexroth, please have the system information ready.

18.2 Frequently Asked Questions

Database and security system

Q. What is saved to the "Distributed Variables" folder?

A. The "Distributed Variables" folder contains tags imported from the PC-based control linked to the "WinStudio" project.

Q. How is it possible to count the number of tags that are configured in the project database?

A. The number of tags currently used in the project is displayed in the status bar at the bottom of the development environment. Each array element and each class element of the tags configured in the WinStudio tag database is counted.

Q. How can the list of "Users" added to the project during runtime with the [CreateUser on page CreateUser](#) function be displayed?

A. Execute the following command:

"WinStudio\BIN\Studio Manager.exe" "WinStudio\BIN\ExtUser.dll"

(for example:

"E:\Rexroth\IndraWorks\WinStudio\BIN\Studio Manager.exe" "E:\Rexroth\IndraWorks\WinStudio\BIN\ExtUser.dll").

This command launches a dialog. The users created by the **CreateUser()** function can be displayed to create or delete users.

Graphics

Q. How is it possible to insert and configure ActiveX objects in a project?

A. To insert an ActiveX object in a project screen, proceed as follows:

1. Open a WinStudio screen in the screen editor and click on **ActiveX Control** in the "WinStudio Screen Tools" dialog.

Troubleshooting and FAQs

2. Select the ActiveX control element to be inserted from the list and click on **OK**. The ActiveX object appears in the screen. (Unregistered ActiveX objects are not available in this list field.)
3. Double-click on the ActiveX object and assign a name to it (enter a value in the **Name** field). The animations and methods list can be viewed by selecting **Methods**. The static properties can be set with the **Properties** button (a detailed description of the object properties can be found in the component documentation provided by the component developer).

There are three functions to access the ActiveX component during runtime:

- **XGet(strName, strProperties)**: Returns the value of the properties **strProperties** from the object **strName**.

The list of properties that can be set to the object can be found in the "Object Properties" dialog of the object with the following syntax:

<Properties Name>(PropGet)

(for example **Color(PropGet)**).

- **XSet(strName, strProperties, Value)**: Writes the value **Value** to the properties **strProperties** of the object **strName**. The list of properties that can be set to the object can be found in the "Object Properties" dialog of the object with the following syntax:

<Properties Name>(PropPut)

(for example **Color(PropPut)**).

- **XRun(strName, strMethod, Parameter1, Parameter2, ..., ParameterN)**: Executes the method **strMethod** from the object **strName** according to the parameters **Parameter1, Parameter2, ..., ParameterN**.

The list of methods available in the object can be found in the "Object Properties" dialog of the object with the syntax

<Method name>(Method)

(for example **OpenFile(Method)**).



Before inserting an ActiveX control element (usually an OCX file) into the project, make sure it has been properly registered on the computer.



The number of parameters set in the "[XRun on page 684](#)" function can vary from 0 up to 255 and depends on the respective ActiveX component. To set the parameters, tags can be used. However, the tag type has to correspond to the type of component parameter (Boolean, Integer, String or Real).

Q. How is it possible to designate a screen that opens each time the project is started?

A. Open the WinStudio Properties dialog, click on the **Viewer** tab and enter the name of the startup screen into the **Startup screen** field.

Q. How is it possible to insert background pictures into the screen?

A. Right-click on the screen and select the option **Screen Attributes** from the context menu. Enable the **Enable Background** checkbox and select the picture format in the combobox next to the label. Copy the picture file to the "Screen" subfolder of the project and give it the name of the screen (**Screen-Name.scr**). With the **Shared image** option, a bitmap file can be copied to the

Troubleshooting and FAQs

"Screen" subfolder and shared with more than one screen. In this case, the bitmap name has to be entered into the **Shared image** field.

Tasks

Q. How is it possible to convert the history trend to an ASCII file?

A. To convert a history trend file to ASCII format, copy the file `"..\WinStudio\bin\hst2txt.exe"` to the path `"\<Project directory>\hst\"`. Alternatively, the "HST2TXT" function can be used in a "math" worksheet to convert binary files into text format automatically without having to use a DOS window.

Q. How is it possible to exchange data with FOX Pro via an ODBC protocol?

A. When exchanging data with a FOX Pro database, it is necessary to set the parameter "UseQuote=0" from the [ODBC] section in the **project_name.app** file.

Q. How is it possible to set a **DATE** field for an ODBC interface with an Oracle package?

A. Configure the "Column" cells in the ODBC worksheet with the syntax **ColumnName.ts** (for example: **MyDate.ts**).

Q. How is it possible to execute a "math" worksheet during startup and another "math" worksheet during project shutdown?

A.

- **Startup:** Execute a "math" worksheet during the startup by creating a "math" worksheet and entering the expression **<TagName>=0** (for example, **StartTag=0**) in the **Execution** field. In the last line of the "math" worksheet, set the value "1" to the **<TagName>** tag. The **<TagName>** tag type is to be **Boolean**.
- **Shutdown:** Instead of executing the **ShutDown()** function directly, execute a "math" worksheet and configure the **ShutDown()** function in the last line of this "math" worksheet.

Communication

Q. How is it possible to set a "communication error" alarm?

A. Configure a tag in the **Write Status** or **Read Status** field of the driver worksheets and configure an alarm whenever this tag is not "0" (zero).

Q. How is it possible to communicate with a Siemens S7-200 PLC without using Prosave software?

A. Siemens S7-200 PLC has a Freeport that can implement any protocol via PLC programming. There is a free PLC software distributed by Siemens that implements Modbus protocol in the PLC Freeport (for further details, contact Siemens support). Using this software in the PLC with the "WinStudio" Modbus driver (MODBU), enables exchange of information between them.

Q. How is it possible to start and stop communication drivers during runtime?

A. There are three functions available to handle the execution of the communication drivers during runtime:

- Start all drivers configured in the project: **StartTask("Driver")**
For example: **StartTask("Driver")**

- Start a specific driver configured in the project:

WinExec("..\WinStudio\bin\Studio Manager.exe" + " " + "..\WinStudio\bin\Driver.dll" + " " + "DriverName")

For example:

WinExec(Asc2Str(34) + "Bin\Studio Manager.exe" + Asc2Str(34) + " " + Asc2Str(34) + "Bin\Driver.dll" + Asc2Str(34) + " " + Asc2Str(34) + "MODBU" + Asc2Str(34))



The **Asc2Str(34)** function is used to add quotation marks for paths that contain space characters.

- Stop a specific driver configured in the project:

EndTask("DriverDriverName")

For example:

EndTask("DriverMODBU")

With the **StartTask(TaskName)** and **EndTask(TaskName)** functions, other tasks can be started or stopped.

For example, **StartTask("Viewer")**, **Endtask("Viewer")**.

Q. What are the parameters of the "WinStudio" DDE Server?

A. "WinStudio" DDE Server and NetDDE Server parameters are listed in the table below:

Comm. type	Application	Topic	Element
Network DDE	\\<Computer Name>\NDDE\$	UNISOFT\$	<TagName>
Local DDE	UNIDDE	DB	<TagName>
Network DDE	\\<Computer Name>\NDDE\$	UNISOFT\$	<TagName>
Local DDE	UNIDDE	DB	<TagName>

Q. How is it possible to exchange data with Excel using NetDDE?

A. NetDDE can be used to exchange data via DDE protocol between networked stations.

- Start the DDE Server module from the development application in the task configuration (**Status** in the WinStudio context menu).
- Run Excel on the remote station.
- Open an Excel worksheet and fill the cells that are to exchange data with "WinStudio", using the following syntax:

=\ computer name \NDDE\$!\UNISOFT\$! tag name

For example:

=\PC\NDDE\$!\UNISOFT\$!second



- When running under Windows NT or Windows2000, it has to be made sure that the services Network DDE and Network DDE DSDM are started. (Use the Services shortcut from the Control Panel to start these services).
- When running under windows 98, the program **<Windows directory>\etdde.exe** has to be executed on both computers (e.g. **c:\Windows\etdde.exe**).

Q. Is the "WinStudio" OPC interface compliant with OPC specification v1.0a or v2.0?

A. The "WinStudio" OPC Client and OPC Server modules are compliant with both OPC specification v1.0a and v2.0.

Q. How is it possible to get errors from Intellution / GE Fanuc iFIX applications?

Troubleshooting and FAQs

A. If the project is communicating with an iFIX application via [TCP/IP on page 351](#), the following key is to be added to the project file (i.e., **project_name.APP**):

[TCP]

SetQualityToBadOnError=1

Afterwards, if the iFIX application generates an error during runtime, the quality of the affected tags in the project is set to BAD. This information can be obtained by [reading the Quality tag field on page 129](#) (i.e., **tagname->Quality**).

General Information

Q. Which operating systems are supported by "Rexroth WinStudio" and CE-View?

A. Refer to the following table. "N" means that the operating system is NOT supported, and "Y" means that the operating system is supported.

Operating system		"Rexroth WinStudio"				CEView			
Name	Version	v2.x	v3.x to v4.1	v4.2 to v6.0	v6.1 or higher	v3.x	v4.x	v5.x	v6.x
Windows Vista	Any	N	N	N	Y	N	N	N	N
Windows XP	Any	N	Y	Y	Y	N	N	N	N
Windows 2000	Any	N	Y	Y	Y	N	N	N	N
Windows NT	v4.0+SP4 or higher	Y	Y	Y	N	N	N	N	N
Windows ME	Any	N	Y	N	N	N	N	N	N
Windows 98	Any	N	Y	N	N	N	N	N	N
Windows 95	Any	Y	Y	N	N	N	N	N	N
Windows CE	v2.12	N	N	N	N	Y	N	N	N
	v3.x	N	N	N	N	Y	Y	Y	Y
Windows CE.Net	v4.0	N	N	N	N	N	N	Y	Y
	v4.1	N	N	N	N	N	N	Y	Y
	v4.2	N	N	N	N	N	N	Y	Y
	v5.0	N	N	N	N	N	N	N	Y
	v6.0	N	N	N	N	N	N	N	Y

Q. How is it possible to start "WinStudio" automatically when the computer is switched on?

A. Create a shortcut to [...]WinStudio\Bin\RunStudio.exe and place it in the Startup folder (C:\WINDOWS\Documents and Settings\All Users\Start Menu\Programs\Startup).

Q. How is it possible to disable Dr. Watson?

A. The step-by-step procedure to disable Dr. Watson under Windows NT is described below:

1. Execute the program <Windows directory>\RegEdit.exe.
(for example,
C:\WinNT\Regedit.exe)

2. Select the path **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AeDebug**
3. Set the value 0 (zero) to the parameter "Auto" from the path selected.
4. Close the "Registry Editor" window.

NOTICE

Special cautions have to be taken when editing parameters in the "Registry Editor" program as some of them can modify the overall behavior of the operating system.

There are two ways to modify the date format in "WinStudio":

- **Offline:** Set the parameters **Order=<DateFormat>** (for example, "DMY") and **Separator=Separator** (for example, "/" or .) from the [International] section in the [...] \WinStudio\Bin\Program Settings.ini file.



The project has to be verified to apply date settings to previously saved Web pages.

- **Online:** Use the function **SetDateFormat(strSeparator,strDateFormat)**. For example, **SetDateFormat("/", "MDY")**.

Q. What features are not supported by CEView?

A. The main features not supported by CEView are: DDE, NetDDE, ODBC, PasteLink, and a number of functions.

Q. How is it possible to display a splash window when starting a project in CEView?

A. To enable the project to display a splash window during startup, add the following key to the **CEView.ini** file of the device:

CEView.ini file

```
[OEM] SpashWnd = Path to bitmap File // default is Splash.bmp  
SplashWndTime = Time in milliseconds // default is 1000
```

Q. How is it possible to access runtime messages and errors on a Windows Embedded device without display?

A. If the device has a network connection, use the [Remote LogWin tool on page 389](#) to view the messages of the device as they are generated.

The messages can be saved to a dump file (**dump.txt**) and accessed if necessary. To create the file, add the following key to the **CEView.ini file of the device: [OEM] BlindDevice = 1**

Q. What are the main steps to create a Web-based application?

A. Proceed as follows:

1. Develop the project locally. Do not use features that are not supported by Thin Clients for the screens that are to be saved in HTML format.
2. After saving the screens in standard format (**Save** in the context menu), save the screens for the Thin Client in HTML format (**Save as HTML** in the context menu).
3. Open the "Web Settings" dialog in the context menu of the "WinStudio" folder.

Troubleshooting and FAQs

4. In the **Data Server IP** field, enter the IP address of the server station (i.e., the station on which the project is running).
5. Open the task configuration ("Status" in the WinStudio context menu) and set the TCP/IP Server module to **Startup=Automatic**.
6. Open the "Application Variables" editor and set the option **Server** instead of **Local** in the **Scope** column for all tags that have to exchange values between the server and the Thin Client station.
7. Verify the project (**Verify** in the WinStudio context menu) to update the HTML files with these new settings.
8. If no web server is running on the computer, copy the program "NTWeb-Server.exe" from the "Bin" subfolder of the program directory into the Web directory (e.g., \project name\Web) and run it. The path where this web server program is executed is the root directory of the server. The web server is required to export data (web files) in HTTP protocol to the Thin Clients.
9. Run the project on the server station.
10. Using a browser (e.g., Internet Explorer) on the Thin Client station, enter the URL to download the screen that was saved in HTML format (for example, <http://ServerIPAddress/ScreenName.html>).



The Thin Client requires an ActiveX component (**ISSymbol.ocx**) to handle the screens on the browser. If the Thin Client is connected to the Internet, this component is downloaded and registered automatically. Otherwise, it is necessary to copy it to the \OSPath\System32 directory of the Thin Client and register it by the command **regsvr32 ISSymbol.ocx**. This file can be found in the \BIN folder in the "WinStudio" installation directory.

Q. How is it possible to maintain communication between a Thin Client connecting via proxy and a Web Gateway application running on Microsoft IIS?

A. Microsoft Internet Information Services (IIS) has a configuration option to maintain HTTP connections. If this option is enabled, it may conflict with Thin Clients that are connecting via proxy. To disable this option, proceed as follows:

1. Start the Internet Services Manager.
2. In the "Internet Information Services" window, open the local server ("*server name").
3. Right-click on **Default Web Sites** and select **Properties** from the context menu. The "Default Web Site Properties" dialog is displayed.
4. Select the **Web Site** tab of the "Default Web Site Properties" dialog.
5. In the "Connections" pane of the **Web Site** tab, disable the **HTTP Keep-Alives Enabled** option.
6. Click on **OK** to save the change and close the dialog.

Q. How is it possible to send an email from the "WinStudio" project?

A. Proceed as follows:

- Execute the function **CNFE-Mail(strSMTP,strFrom,strPOP3,strUser,strPassword,numTimeOut)** to configure the overall parameters used to send emails. After executing this function, the set parameters are kept in the system until the project is shut down. For this reason, most projects execute this function just once after they are started:

Troubleshooting and FAQs

- Execute the function

SendEmail(strSubject,strMessage,strTO) and/or SendEmailExt(strSubject,strMessage,strTO,strCC,strBCC, strFile1,...,strFileN)

each time an email message is to be sent. The main differences between both functions are listed in the following table:

Property	SendEmail	SendEmailExt
Execution	synchronous	asynchronous
Supports Subject text	Y	Y
Supports Message text	Y	Y
Supports TO addresses	Y	Y
Supports CC addresses	N	Y
Supports BCC addresses	N	Y
Supports attached files	N	Y

Q. The runtime task (TCP/IP, OPC, DDE, ODBC, etc.) does not work.

A. Ensure that the runtime task is set to **Automatic** in the task configuration (**Status** in the WinStudio folder context menu). Select a runtime task that has to be executed (e.g., TCP/IP Server), click on **Startup** and set it to **Automatic**.

Q. The browser on the Thin Client does not display the screen and launches a warning message regarding ISSymbol.ocx.

A. Ensure that the runtime task is set to **Automatic** in the "Execution Tasks" dialog (**Tasks** in the WinStudio folder context menu). Select a runtime task that has to be executed (e.g., TCP/IP Server), click on **Startup** and set it to **Automatic**.

Q. The browser on the Thin Client launches an error message missing the **ISSymbol.ocx** and does not display the screens of the server.

A. **ISSymbol.ocx** is the ActiveX object used by the browser on the Thin Client to view the web pages. If the Thin Client is connected to the Internet, the **ISSymbol.ocx** control is automatically downloaded and registered on the Thin Client station. Otherwise, it has to be copied to the **WinNT\System32** folder of the Thin Client station and registered manually. Once registered, the browser is able to view the pages.



Use the **regsvr32 ISSymbol32.ocx** command to register the ActiveX component on the Thin Client.

Q. The screens are displayed on the Thin Client (browser), however, the data (tag values) is not read out by the server.

A. Ensure that the parameter in the **Scope** column in the "Application Variables" folder is set to **Server** instead of **Local**. The tags set to **Server** maintain the same value in the Server and in the Thin Client (Browser). The tags set to **Local** have independent values in the Server and in the Thin Client (Browser).

NOTICE

After changing the tag settings, the project has to be verified. Otherwise, the changes are not updated on the web pages.

Troubleshooting and FAQs

Q. The "On Up" expressions configured in the command animation are not executed.

A. The "On Up" expressions from the command animation are not executed if the cursor is dragged out of the object area before releasing it. If the checkbox **Release from the Command Object Properties** window is enabled, the On Up expression is executed even if the cursor is moved out of the object area before releasing it.

Q. The Trend History does not work after adding or removing tags in the "Trend" worksheet.

A. If a tag is inserted into or removed from a "Trend" worksheet, the format of the history files (*.hst) is modified. The same *.hst file cannot have two different formats. Otherwise, the data cannot be retrieved from it properly by the trend object. There are two ways of adding tags to or removing tags from history files: Create a new "Trend" worksheet or delete the old *.hst files.

Q. The value of indirect tags (@TagName) is not shown in the Thin Client program.

A. If a screen is saved as HTML, "WinStudio" saves a <ScreenName>.tagl file in the "\WEB" subfolder. This file contains a list of all tags configured in the screen (objects and animations). If a screen is opened in the Thin Client browser, the tags listed in the <ScreenName>.tagl are "enabled" for TCP/IP communication with the server station. This way, an optimized communication between the server station and the Thin Client stations is provided.

When using indirect tags in this way (@IndirectTag), the displayed tags do not exchange data with the server, unless they have been configured in the screen. In other words, the tags to be pointed to in the screen have to be configured in an screen object to enable TCP/IP communication for these tags with the server station.



Add a transparent rectangle (no fill and no line) in the corner of the screen. Apply the command animation to this rectangle and configure the tags (which can be pointed to by indirect tags during runtime in the Thin Client station) in the **Expression** fields (keep the **Variable Name** fields blank). These tags are added to the <ScreenName>.TAGL file and are available for TCP/IP communication with the server station.

Q. Which functionalities are not supported by PocketPC platforms (for example, IPaq, Cassiopeia, Jornada)?

A. PocketPC devices with Windows CE do not support some of the functionality that is supported by Windows CE devices with the "standard" Windows CE version:

Functionality not supported by PocketPC devices
DCOM (Distributed Component Object Model): All features based on DCOM (e.g., remote OPC communication) are not supported by PocketPC devices.
The function DialGetClientIP on page 670 does not work on PocketPC devices.

19 Appendix: WinStudio-Internal Scripting Language

19.1 Appendix: Internal Scripting Language, General Information

This section describes all functions of the internal WinStudio scripting language. Each functional description includes complete syntax, possible output values, and examples of usage.

19.2 List of Available Functions

The functions described in the following are available in "Rexroth WinStudio".

Log message

Function	Execution	Windows PC	Windows CE	Thin Client
Trace, page 483	synchronous	supported	supported	not supported

Arithmetic

Function	Execution	Windows PC	Windows CE	Thin Client
Abs, page 485	synchronous	supported	supported	supported
Div, page 485	synchronous	supported	supported	supported
Format, page 486	synchronous	supported	supported	supported
GetBit, page 489	synchronous	supported	supported	supported
Mod, page 490	synchronous	supported	supported	supported
Pow, page 490	synchronous	supported	supported	supported
ResetBit, page 491	synchronous	supported	supported	supported
Round, page 492	synchronous	supported	supported	supported
SetBit, page 492	synchronous	supported	supported	supported
SQRT, page 493	synchronous	supported	supported	supported
Swap16, page 493	synchronous	supported	supported	supported
Swap32, page 494	synchronous	supported	supported	supported
Trunc, page 494	synchronous	supported	supported	supported

Statistical

Function	Execution	Windows PC	Windows CE	Thin Client
Avg, page 495	synchronous	supported	supported	supported
Max, page 496	synchronous	supported	supported	supported
Min, page 497	synchronous	supported	supported	supported
Rand, page 498	synchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Logarithmic

Function	Execution	Windows PC	Windows CE	Thin Client
Exp, page 498	synchronous	supported	supported	supported
Log, page 499	synchronous	supported	supported	supported
Log10, page 499	synchronous	supported	supported	supported

Logical

Function	Execution	Windows PC	Windows CE	Thin Client
False, page 501	synchronous	supported	supported	supported
If, page 501	synchronous	supported	supported	supported
Toggle, page 502	synchronous	supported	supported	supported
True, page 503	synchronous	supported	supported	supported

Character string

Function	Execution	Windows PC	Windows CE	Thin Client
Asc2Str, page 504	synchronous	supported	supported	supported
CharToValue, page 505	synchronous	supported	supported	supported
CharToValueW, page 505	synchronous	supported	supported	supported
ClassMembersToStrVector, page 506	synchronous	supported	supported	supported
NCopy, page 507	synchronous	supported	supported	supported
Num, page 508	synchronous	supported	supported	supported
Str, page 508	synchronous	supported	supported	supported
Str2Asc, page 509	synchronous	supported	supported	supported
StrCompare, page 509	synchronous	supported	supported	supported
StrCompareNoCase, page 510	synchronous	supported	supported	supported
StrFromInt, page 510	synchronous	supported	supported	supported
StrFromReal, page 511	synchronous	supported	supported	supported
StrFromTime, page 512	synchronous	supported	supported	supported
StrGetElement, page 513	synchronous	supported	supported	supported
StrLeft, page 513	synchronous	supported	supported	supported
StrLen, page 514	synchronous	supported	supported	supported
StrLower, page 514	synchronous	supported	supported	supported
StrRChr, page 514	synchronous	supported	supported	supported
StrRight, page 515	synchronous	supported	supported	supported
StrSetElement, page 515	synchronous	supported	supported	supported
StrStr, page 516	synchronous	supported	supported	supported
StrStrPos, page 516	synchronous	supported	supported	supported
StrTrim, page 517	synchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Function	Execution	Windows PC	Windows CE	Thin Client
StrTrimAll, page 518	synchronous	supported	supported	supported
StrUpper, page 518	synchronous	supported	supported	supported
ValueToChar, page 519	synchronous	supported	supported	supported
ValueWToChar, page 519	synchronous	supported	supported	supported

Date & Time

Function	Execution	Windows PC	Windows CE	Thin Client
ClockGetDate, page 520	synchronous	supported	supported	supported
ClockGetDayOfWeek, page 521	synchronous	supported	supported	supported
ClockGetTime, page 522	synchronous	supported	supported	supported
DateTime2Clock, page 522	synchronous	supported	supported	supported
GetClock, page 523	synchronous	supported	supported	supported
Hour2Clock, page 523	synchronous	supported	supported	supported
SetSystemDate, page 524	synchronous	supported	supported	supported
SetSystemTime, page 524	synchronous	supported	supported	supported

Trigonometric

Function	Execution	Windows PC	Windows CE	Thin Client
ACos, page 525	synchronous	supported	supported	supported
ASin, page 525	synchronous	supported	supported	supported
ATan, page 526	synchronous	supported	supported	supported
Cos, page 526	synchronous	supported	supported	supported
Cot, page 526	synchronous	supported	supported	supported
Pi, page 527	synchronous	supported	supported	supported
Sin, page 527	synchronous	supported	supported	supported
Tan, page 527	synchronous	supported	supported	supported

Opening and Closing Windows

Function	Execution	Windows PC	Windows CE	Thin Client
Close, page 528	asynchronous	supported	supported	not supported
Open, page 529	asynchronous	supported	supported	supported
OpenPrevious, page 531	asynchronous	supported	supported	supported
ShowInplaceInput, page 532	asynchronous	supported	supported	supported
ShowMessageBox, page 533	asynchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Security

Function	Execution	Windows PC	Windows CE	Thin Client
BlockUser, page 535	synchronous	supported	supported	supported
CheckESign, page 535	synchronous	supported	supported	supported
CreateUser, page 536	synchronous	supported	supported	supported
ExportSecuritySystem, page 537	synchronous	supported	supported	supported
GetSecuritySystemStatus, page 538	synchronous	supported	supported	supported
GetUserFullName, page 539	synchronous	supported	supported	supported
GetUserNames, page 540	synchronous	supported	supported	executed on the server
GetUserPWDAging, page 541	synchronous	supported	supported	supported
GetUserState, page 541	synchronous	supported	supported	supported
RemoveUser, page 543	synchronous	supported	supported	supported
ImportSecuritySystem, page 542	synchronous	supported	supported	supported
SetPassword, page 544	synchronous	supported	supported	supported
UnblockUser, page 545	synchronous	supported	supported	supported

Module activity

Function	Execution	Windows PC	Windows CE	Thin Client
AppActivate, page 547	asynchronous	supported	supported	supported
AppIsRunning, page 548	synchronous	supported	supported	supported
AppPostMessage, page 549	synchronous	supported	supported	supported
AppSendKeys, page 550	synchronous	supported	not supported	supported
CleanReadQueue, page 550	synchronous	supported	not supported	executed on the server
CloseSplashWindow, page 550	synchronous	supported	not supported	executed on the server
DisableMath, page 551	asynchronous	supported	supported	executed on the server
EnableMath, page 551	asynchronous	supported	supported	executed on the server
EndTask, page 551	asynchronous	supported	supported	executed on the server
ExitWindows, page 552	asynchronous	supported	not supported	supported
IsScreenOpen, page 553	synchronous	supported	supported	not supported
IsTaskRunning, page 553	synchronous	supported	supported	executed on the server
IsViewerInFocus, page 554	synchronous	supported	not supported	not supported
KeyPad, page 554	asynchronous	supported	supported	supported
LogOff, page 556	asynchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Function	Execution	Windows PC	Windows CE	Thin Client
LogOn, page 556	asynchronous	supported	supported	supported
Math, page 557	synchronous	supported	supported	executed on the server
PostKey, page 557	synchronous	supported	supported	supported
Recipe, page 558	synchronous	supported	supported	executed on the server
Report, page 558	synchronous	supported	supported	executed on the server
RunGlobalProcedureOnServer, page 564	synchronous	supported	supported	supported
RunVBScript, page 566	synchronous	supported	supported	supported
SecureViewerReload, page 567	synchronous	not supported	not supported	supported (Secure Viewer only)
SendKeyObject, page 568	synchronous	supported	not supported	not supported
SetAppPath, page 570	synchronous	supported	not supported	executed on the server
SetViewerInFocus, page 570	synchronous	supported	not supported	supported
SetViewerPos, page 571	synchronous	supported	not supported	supported
ShutDown, page 572	synchronous	supported	supported	not supported
StartTask, page 572	asynchronous	supported	supported	executed on the server
ViewerPostMessage, page 573	asynchronous	supported	supported	supported
Wait, page 574	synchronous	supported	supported	supported
WinExec, page 574	asynchronous	supported	supported	supported
WinExecIsRunning, page 576	synchronous	supported	supported	supported

File

Function	Execution	Windows PC	Windows CE	Thin Client
DeleteOlderFiles, page 577	synchronous	supported	supported	supported
DirCreate, page 578	synchronous	supported	supported	supported
DirDelete, page 579	synchronous	supported	supported	supported
DirLength, page 579	synchronous	supported	supported	supported
DirRename, page 580	synchronous	supported	supported	supported
FileCopy, page 581	synchronous	supported	supported	supported
FileDelete, page 581	synchronous	supported	supported	supported
FileLength, page 582	synchronous	supported	supported	supported
FileRename, page 582	synchronous	supported	supported	supported
FindFile, page 584	synchronous	supported	supported	supported
FileWrite, page 583	synchronous	supported	supported	supported
FindPath, page 585	synchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Function	Execution	Windows PC	Windows CE	Thin Client
GetFileAttributes, page 585	synchronous	supported	supported	supported
GetFileTime, page 586	synchronous	supported	supported	supported
GetHstInfo, page 586	synchronous	supported	supported	supported
GetLine, page 587	synchronous	supported	supported	supported
HST2TXT, page 590	asynchronous	supported	supported	supported
HST2TXTIsRunning, page 591	synchronous	supported	not supported	executed on the server
PDFCreate, page 594	synchronous	supported	not supported	supported
Print, page 595	asynchronous	supported	supported	supported
RDFileN, page 595	synchronous	supported	supported	supported
WebGetFile, page 596	synchronous	supported	supported	supported

Graphical

Function	Execution	Windows PC	Windows CE	Thin Client
AutoFormat, page 597	synchronous	supported	supported	supported
GetScrInfo, page 598	synchronous	supported	supported	supported
PrintSetup, page 599	asynchronous	supported	supported	supported
PrintWindow, page 599	asynchronous	supported	supported	not supported
ResetDecimalPointsTable, page 601	synchronous	supported	supported	supported
RGBColor, page 601	synchronous	supported	supported	supported
RGBComponent, page 602	synchronous	supported	supported	supported
SetDecimalPoints, page 604	synchronous	supported	supported	supported
SetDisplayUnit, page 605	synchronous	supported	supported	supported
SetTagDisplayUnit, page 606	synchronous	supported	supported	supported

Translation

Function	Execution	Windows PC	Windows CE	Thin Client
Ext, page 607	synchronous	supported	supported	supported
SetLanguage, page 607	synchronous	supported	supported	supported
SetTranslationFile, page 608	synchronous	supported	supported	supported

Multimedia

Function	Execution	Windows PC	Windows CE	Thin Client
Play, page 609	Synchronous/ asynchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

System info

Function	Execution	Windows PC	Windows CE	Thin Client
DbVersion, page 611	synchronous	supported	supported	supported
GetAppHorizontalResolution, page 611	synchronous	supported	not supported	executed on the server
GetAppPath, page 612	synchronous	supported	supported	executed on the server
GetAppVerticalResolution, page 612	synchronous	supported	not supported	executed on the server
GetComputerIP, page 612	synchronous	supported	supported	supported
GetComputerName, page 613	synchronous	supported	not supported	supported
GetCursorX, page 613	synchronous	supported	supported	supported
GetCursorY, page 613	synchronous	supported	supported	supported
GetDisplayHorizontalResolution, page 614	synchronous	supported	supported	supported
GetDisplayVerticalResolution, page 614	synchronous	supported	supported	supported
GetFreeMemoryCE, page 614	synchronous	not supported	supported	supported
GetHardKeyModel, page 615	synchronous	supported	not supported	executed on the server
GetHardKeySN, page 615	synchronous	supported	not supported	executed on the server
GetIPAll, page 616	synchronous	supported	supported	supported
GetMemoryCE, page 616	synchronous	not supported	supported	supported
GetNetMACID, page 617	synchronous	supported	supported	supported
GetOS, page 618	synchronous	supported	supported	supported
GetPrivateProfileString, page 618	synchronous	supported	supported	supported
GetProductPath, page 619	synchronous	supported	supported	supported
GetRegValue, page 619	synchronous	supported	supported	not supported
GetRegValueType, page 620	synchronous	supported	supported	not supported
GetServerHostName, page 621	synchronous	not supported	not supported	supported
GetTickCount, page 621	synchronous	supported	supported	supported
InfoAppAlrDir, page 622	synchronous	supported	supported	supported
InfoAppHSTDir, page 622	synchronous	supported	supported	supported
InfoDiskFree, page 622	synchronous	supported	not supported	supported
InfoResources, page 623	synchronous	supported	supported	supported
IsActiveXReg, page 623	synchronous	supported	supported	supported
IsAppChangedOnServer, page 624	synchronous	supported	supported	supported
NoInputTime, page 625	synchronous	supported	supported	supported
ProductVersion, page 625	synchronous	supported	supported	supported
RegSaveCE, page 626	synchronous	not supported	supported	not supported

Appendix: WinStudio-Internal Scripting Language

Function	Execution	Windows PC	Windows CE	Thin Client
ReloadAppFromServer, page 626	synchronous	supported	supported	supported
SaveAlarmFile, page 627	synchronous	supported	supported	not supported
SetAppAlarmPath, page 628	synchronous	supported	supported	executed on the server
SetAppHSTPath, page 628	synchronous	supported	supported	executed on the server
SetDateFormat, page 629	synchronous	supported	supported	supported
SetKeyboardLanguage, page 629	synchronous	supported	supported	supported
SetRegValue, page 630	synchronous	supported	supported	not supported
SetWebConfig, page 631	synchronous	supported	supported	executed on the server
SNMPGet, page 633	synchronous	supported	not supported	supported
SNMPSet, page 634	synchronous	supported	not supported	supported
WritePrivateProfileString, page 636	synchronous	supported	supported	supported

Tags database

Function	Execution	Windows PC	Windows CE	Thin Client
ExecuteAlarmAck, page 637	synchronous	supported	supported	supported
ForceTagChange, page 638	synchronous	supported	supported	supported
GetTagValue, page 638	synchronous	supported	supported	supported
SetTagValue, page 639	synchronous	supported	supported	supported

Loop

Function	Execution	Windows PC	Windows CE	Thin Client
For, page 640 & "Next"	N/A	supported	supported	not supported

ODBC

Function	Execution	Windows PC	Windows CE	Thin Client
ODBCBeginTrans, page 642	synchronous	supported	not supported	executed on the server
ODBCBindCol, page 642	synchronous	supported	not supported	executed on the server
ODBCCanAppend, page 644	synchronous	supported	not supported	executed on the server
ODBCCanTransact, page 644	synchronous	supported	not supported	executed on the server
ODBCCanUpdate, page 644	synchronous	supported	not supported	executed on the server
ODBCClose, page 645	synchronous	supported	not supported	executed on the server

Appendix: WinStudio-Internal Scripting Language

Function	Execution	Windows PC	Windows CE	Thin Client
ODBCCommitTrans, page 645	synchronous	supported	not supported	executed on the server
ODBCDelete, page 646	synchronous	supported	not supported	executed on the server
ODBCExecuteSQL, page 646	synchronous	supported	not supported	executed on the server
ODBCInsert, page 647	synchronous	supported	not supported	executed on the server
ODBCIsBOF, page 648	synchronous	supported	not supported	executed on the server
ODBCIsDeleted, page 648	synchronous	supported	not supported	executed on the server
ODBCIsEOF, page 649	synchronous	supported	not supported	executed on the server
ODBCIsFieldNULL, page 649	synchronous	supported	not supported	executed on the server
ODBCIsFieldNullable, page 650	synchronous	supported	not supported	executed on the server
ODBCMove, page 650	synchronous	supported	not supported	executed on the server
ODBCMoveFirst, page 651	synchronous	supported	not supported	executed on the server
ODBCMoveLast, page 651	synchronous	supported	not supported	executed on the server
ODBCMoveNext, page 652	synchronous	supported	not supported	executed on the server
ODBCMovePrev, page 652	synchronous	supported	not supported	executed on the server
ODBCOpen, page 653	synchronous	supported	not supported	executed on the server
ODBCQuery, page 654	synchronous	supported	not supported	executed on the server
ODBCRollback, page 654	synchronous	supported	not supported	executed on the server
ODBCSetFieldNULL, page 655	synchronous	supported	not supported	executed on the server
ODBCSetFilter, page 656	synchronous	supported	not supported	executed on the server
ODBCSetSort, page 656	synchronous	supported	not supported	executed on the server
ODBCUnbindCol, page 657	synchronous	supported	not supported	executed on the server
ODBCUpdate, page 657	synchronous	supported	not supported	executed on the server

Appendix: WinStudio-Internal Scripting Language

Email

Function	Execution	Windows PC	Windows CE	Thin Client
CnfEmail, page 658	synchronous	supported	supported	supported
GetStatusSendEmailExt, page 661	synchronous	supported	supported	supported
SendEmail, page 661	synchronous	supported	supported	supported
SendEmailExt, page 662	asynchronous	supported	supported	supported

Dial-up

Function	Execution	Windows PC	Windows CE	Thin Client
DialError, page 664	synchronous	supported (see note)	supported	supported
DialGetClientIP, page 670	synchronous	supported (see note)	supported	supported
DialGetServerIP, page 672	synchronous	supported (see note)	not supported	supported
DialStatus, page 673	synchronous	supported (see note)	supported	supported
DialUp, page 675	asynchronous	supported (see note)	supported	supported
DialUpToCE, page 676	asynchronous	supported (see note)	not supported	supported
FindAllDevices, page 678	synchronous	supported (see note)	supported	supported
FindModem, page 678	synchronous	supported (see note)	supported	supported
HangUp, page 679	synchronous	supported (see note)	supported	supported
PhoneDialUp, page 680	asynchronous	supported (see note)	not supported	supported
PhoneDisableListen, page 680	synchronous	supported (see note)	not supported	supported
PhoneEnableListen, page 681	synchronous	supported (see note)	not supported	supported
PhoneHangUp, page 682	asynchronous	supported (see note)	not supported	supported
PhoneStatus, page 682	synchronous	supported (see note)	not supported	supported

The Dial-Up functions are not supported on Windows 7.

Appendix: WinStudio-Internal Scripting Language

ActiveX control and .NET control

Function	Execution	Windows PC	Windows CE	Thin Client
XGet, page 684	asynchronous	supported	ActiveX controls only	supported
XRun, page 684	asynchronous	supported	ActiveX controls only	supported
XSet, page 685	asynchronous	supported	ActiveX controls only	supported

Event logger

Function	Execution	Windows PC	Windows CE	Thin Client
SendEvent, page 686	synchronous	supported	supported	supported

FTP

Function	Execution	Windows PC	Windows CE	Thin Client
CnfFTP, page 687	synchronous	supported	supported	supported
FTPGet, page 688	asynchronous	supported	supported	supported
FTPPut, page 690	asynchronous	supported	supported	supported
FTPStatus, page 691	synchronous	supported	supported	supported

DB/ERP

Function	Execution	Windows PC	Windows CE	Thin Client
DBCursorClose, page 693	synchronous	supported	supported	supported
DBCursorCurrentRow, page 695	synchronous	supported	supported	supported
DBCursorGetValue, page 696	synchronous	supported	supported	supported
DBCursorMoveTo, page 697	synchronous	supported	supported	supported
DBCursorNext, page 698	synchronous	supported	supported	supported
DBCursorOpen, page 699	synchronous	supported	supported	supported
DBCursorOpenSQL, page 700	synchronous	supported	supported	supported
DBCursorPrevious, page 702	synchronous	supported	supported	supported
DBCursorRowCount, page 703	synchronous	supported	supported	supported
DBDelete, page 703	synchronous	supported	supported	supported
DBExecute, page 704	synchronous	supported	supported	supported
DBInsert, page 706	synchronous	supported	supported	supported
DBSelect, page 707	synchronous	supported	supported	supported
DBUpdate, page 709	synchronous	supported	supported	supported
SyncAlarm, page 710	asynchronous	supported	supported	executed on the server
SyncAlarmStatus, page 711	synchronous	supported	supported	executed on the server

Appendix: WinStudio-Internal Scripting Language

Function	Execution	Windows PC	Windows CE	Thin Client
SyncEvent, page 711	asynchronous	supported	supported	executed on the server
SyncEventStatus, page 712	synchronous	supported	supported	executed on the server
SyncTrend, page 713	asynchronous	supported	supported	executed on the server
SyncTrendStatus, page 714	synchronous	supported	supported	executed on the server

19.3 Structure of the Functional Descriptions

This is an explanation of the description of the internal scripting functions.

The functional descriptions are divided into several sections.

Function attributes

Each function has certain attributes that are described in a single-line table:

Function	Group	Execution	Windows	Embedded	Thin Client
Function name	Group name	synchronous or asynchronous	supported or not supported	supported or not supported	supported or not supported

The first column specifies the exact name of the function as it is to be used in the project.

The second column specifies the name of the **group** the function is assigned to. Each group combines functions according to the type of calculation or the part of the project it affects. The group name supports searching for a specific function in the function browser or the documentation.

The **Execution** column describes whether the function is executed synchronously or asynchronously:

- **Synchronous** means that the respective station requires a response or confirmation by the remote station if the function is executed on the project server or the project client. If a synchronous function is requested, the runtime environment waits until the function has been processed. This way, the execution remains synchronous on server and client. This is no problem if functions are executed very quickly. However, if a client executes many functions or the network is slow, the performance of the application may be affected.
- **Asynchronous** means that the function can be executed on either the project server or the project client and the executing function does not have to wait for the result. Thus, the project continues to run without interruption.

The following columns contain information on which target systems support the function:

- The **Windows** column provides information on whether the function can be executed on client or server stations that are running on a full desktop or server version of Microsoft Windows.
- The **Embedded** column provides information on whether the function can be executed on client and server stations under Windows CE.
- The **Client** column provides information on whether the function can be executed on client stations with Secure Viewer or a web browser.

Appendix: WinStudio-Internal Scripting Language

For further information about these target systems, refer to [System requirements on page 37](#).

Syntax diagram and parameters

Syntax diagrams show how the function is to be entered and what parameters it requires.

In most cases, parameters can either contain a constant value or a variable (tag). The data type of the parameter is indicated by its prefix:

- "bool" indicates that the parameter requires a Boolean value or a Boolean variable.

For example, either "0" or "boolTag".

- "num" indicates that the parameter requires a numeric value or an Integer or Real variable.

For example, either "45.6543" or "numTag".

- "str" indicates that the parameter requires a string in quotation marks or a text variable (String tag).

For example, either ""My string"" or "strTag".

The additional prefix "opt" indicates that a parameter is optional. If no value is specified for the parameter, the function takes the default value specified in the parameter description.

If a parameter requires a variable or other special input, this is explained in detail in the parameter description.

Returned value

This section describes the value that is potentially returned by the function.

This can be the result of a calculation or an error code.

An error code indicates whether a function was executed without errors or helps to determine the cause of occurring problems.

Notes

This section describes additional instructions or warnings on the use of the function.

Examples

This section contains code examples of how the function can be executed. Various examples illustrate how constant values, variables and optional parameters can be used to execute the function.

19.4 Log Message Functions

19.4.1 Log Message Functions – General Information

These functions are used to display status and error messages in the "Output" (for local runtime) or "Remote LogWin" window.

19.4.2 Trace

"Trace" is an internal scripting function that displays a text message in the "Output" window. It is typically used to trace the program sequence.

Function	Group	Execution	Windows	Embedded	Thin Client
Trace	Log message	synchronous	supported	supported	not supported

Syntax

Trace(strOutputMessage)

Inputs

strOutputMessage

The text of the message to be displayed.

Appendix: WinStudio-Internal Scripting Language

Examples

Display static text that reports a specific event

```
Trace("Beginning step 5.")
```

Display a date or time stamp by referencing the appropriate system tag

```
Trace(Date)
```

Link static text, tag references, and function calls to generate a complex message

```
Trace("The current second of the minute is " + Second + " and the system tick is " +  
GetTickCount() + " ms.")
```

19.5 Arithmetic Functions

19.5.1 Arithmetic Functions – General Information

These functions are used to perform arithmetic operations and bit manipulation on numeric values.

Arithmetic

Function	Execution	Windows PC	Windows CE	Thin Client
Abs, page 485	synchronous	supported	supported	supported
Div, page 485	synchronous	supported	supported	supported
Format, page 486	synchronous	supported	supported	supported
GetBit, page 489	synchronous	supported	supported	supported
Mod, page 490	synchronous	supported	supported	supported
Pow, page 490	synchronous	supported	supported	supported
ResetBit, page 491	synchronous	supported	supported	supported
Round, page 492	synchronous	supported	supported	supported
SetBit, page 492	synchronous	supported	supported	supported
SQRT, page 493	synchronous	supported	supported	supported
Swap16, page 493	synchronous	supported	supported	supported
Swap32, page 494	synchronous	supported	supported	supported
Trunc, page 494	synchronous	supported	supported	supported

19.5.2 Arithmetic Operators

The internal scripting language supports the following arithmetic operators.

Arithmetic operators

Operator	Usage	Description
+	X + Y	Add (plus)
-	X - Y	Subtract (minus)
*	X * Y	Multiply by
/	X / Y	Divide by

Arithmetic operators are resolved from left to right according to the standard order of operations. To change the order, the part of the equation to be re-

Appendix: WinStudio-Internal Scripting Language

solved first is to be enclosed in parentheses. For example, the following equation produces a result of 11 as multiplication is carried out before addition. The equation multiplies 2 by 3 and then adds 5 to the result:

$$5+2*3$$

In contrast, if parentheses are used to change the syntax, 5 and 2 are added and then multiplied by 3 to produce a result of 21:

$$(5+2)*3$$



For more complex mathematical operations, use the [Arithmetic on page 484](#), [Statistical on page 495](#), [Logarithmic on page 498](#) and [Trigonometric on page 524](#) functions.

19.5.3 Abs

"Abs" is an internal scripting function that gets the absolute value of a specified numeric value.

Function	Group	Execution	Windows	Embedded	Thin Client
Abs	Arithmetic	synchronous	supported	supported	supported

Syntax

Abs(numValue)

Inputs

numValue

The numeric value out of which the function determines the absolute value.

Returned value

The absolute value of the specified numeric value.

Examples

Tag name	Expression
Tag	ABS("-54.9788") // Returned value = 54.9788
Tag	ABS(numValue) // Returned value = absolute value of the number in the numValue tag.

19.5.4 Div

"Div" is an internal scripting function to calculate the quotient of two specified numeric values.

Function	Group	Execution	Windows	Embedded	Thin Client
Div	Arithmetic	synchronous	supported	supported	supported

Syntax

Div(numNumerator, numDenominator)

Inputs

numNumerator

The numerator of the division (dividend).

Appendix: WinStudio-Internal Scripting Language

numDenominator

The denominator of the division (divisor).

Returned value This function returns the quotient only as a whole number. The remainder is omitted.



Use the [Mod page 490](#) function to get the remainder instead of the quotient.

Examples

Tag name	Expression
numValue	Div(100, 8) // Returns the value 12
numValue	Div(16, 4) // Returns the value 4
numValue	Div(100, 12.5) // Returns the value 8

19.5.5 Format

"Format" is an internal scripting function that formats a numeric value and returns it as a string.

Function	Group	Execution	Windows	Embedded	Thin Client
Format	Arithmetic	synchronous	supported	supported	supported

Syntax

```
Format(strFlag, numValue)
```

Inputs**strFlag**

A description of how the given numeric value is to be formatted according to the syntax

%width.precisionFormat , where:

- width** is the minimum number of characters to be returned by the function. If the value to be returned is shorter than this, it is padded either with blank spaces (" ") or zeroes ("0"). Refer to the following examples.
The value is not shortened even if the result is longer than the specified width.
Applicable for flags **d, x, X, o, b, f, e, E, g, G, s, c** and **h**.
- .precision** is the number of decimal places for a floating-point number.
Applicable for flags **f, e, E, g** and **G**.
- F** is the specific format:

Format	Description
d	Decimal
x	Hexadecimal (alphabetic characters in lowercase)
X	Hexadecimal (alphabetic characters in uppercase)
o	Octal
b	Binary
f	Floating-point

Appendix: WinStudio-Internal Scripting Language

Format	Description
e	Scientific notation (e in lowercase)
E	Scientific notation (E in uppercase)
g	Rounded (e in lowercase, if applicable)
G	Rounded (E in uppercase, if applicable)
s	Character string
c	ASCII character (i.e., the numeric value is interpreted as an ASCII character code)
h	Hours (hh:mm:ss)

Alternatively, the format can be set using the syntax `"##.###"`, where the numeric value is rounded to the specified number of decimal places.

numValue

The numeric value to be formatted.

Returned value This function returns a string that contains the formatted numeric value.



"Format" is similar to the "printf" function in other programming languages and provides most similar formatting options. However, unlike "printf", "Format" can be used to format only one numeric value at a time.

This function is particularly useful for formatting values to be output in reports.

Examples

Tag name	Expression
Tag	Format("%d", 12.34) // Returned value = "12"
Tag	Format("%04d", 12.34) // Returned value = "0012"
Tag	Format("%4d", 12.34) // Returned value = "12"

Tag name	Expression
Tag	Format("%x", 26) // Returned value = "1a"
Tag	Format("%04x", 26) // Returned value = "001a"
Tag	Format("%4x", 26) // Returned value = "1a"

Tag name	Expression
Tag	Format("%X", 26) // Returned value = "1A"
Tag	Format("%04X", 26) // Returned value = "001A"
Tag	Format("%4X", 26) // Returned value = "1A"

Tag name	Expression
Tag	Format("%o", 16) // Returned value = "20"
Tag	Format("%04o", 16) // Returned value = "0020"
Tag	Format("%4o", 16) // Returned value = "20"

Appendix: WinStudio-Internal Scripting Language

Tag name	Expression
Tag	Format("%b", 2) // Returned value = "10"
Tag	Format("%4b", 2) // Returned value = "0010"
Tag	Format("%04b", 2) // Returned value = "0010"

Tag name	Expression
Tag	Format("%0.1f", 12.34) // Returned value = "12.3"
Tag	Format("%06.1f", 12.34) // Returned value = "0012.3"
Tag	Format("%6.1f", 12.34) // Returned value = "12.3"

Tag name	Expression
Tag	Format("%e", 12.34) // Returned value = "1.234000e+001"
Tag	Format("%0.1e", 12.34) // Returned value = "1.2e+001"
Tag	Format("%09.1e", 12.34) // Returned value = "01.2e+001"
Tag	Format("%9.1e", 12.34) // Returned value = " 1.2e+001"

Tag name	Expression
Tag	Format("%E", 12.34) // Returned value = "1.234000E+001"
Tag	Format("%0.1E", 12.34) // Returned value = "1.2E+001"
Tag	Format("%09.1E", 12.34) // Returned value = "01.2E+001"
Tag	Format("%9.1E", 12.34) // Returned value = " 1.2E+001"

Tag name	Expression
Tag	Format("%0.1g", 12.34) // Returned value = "1e+001"
Tag	Format("%0.2g", 12.34) // Returned value = "12"
Tag	Format("%0.3g", 12.34) // Returned value = "12.3"
Tag	Format("%05.3g", 12.34) // Returned value = "012.3"
Tag	Format("%5.3g", 12.34) // Returned value = " 12.3"

Tag name	Expression
Tag	Format("%0.1G", 12.34) // Returned value = "1E+001"
Tag	Format("%0.2G", 12.34) // Returned value = "12"
Tag	Format("%0.3G", 12.34) // Returned value = "12.3"
Tag	Format("%05.3G", 12.34) // Returned value = "012.3"
Tag	Format("%5.3G", 12.34) // Returned value = "12.3"

Appendix: WinStudio-Internal Scripting Language

Tag name	Expression
Tag	Format("%s", 12.34) // Returned value = "12"
Tag	Format("%04s", 12.34) // Returned value = "0012"
Tag	Format("%4s", 12.34) // Returned value = "12"

Tag name	Expression
Tag	Format("%c", 97) // Returned value = "a"
Tag	Format("%4c", 97) // Returned value = "a"
Tag	Format("%04c", 97) // Returned value = "000a"

Tag name	Expression
Tag	Format("%h", 30) // Returned value = "00:00:30"
Tag	Format("%h", 60) // Returned value = "00:01:00"
Tag	Format("%h", 90) // Returned value = "00:01:30"
Tag	Format("%h", 3600) // Returned value = "01:00:00"

Tag name	Expression
Tag	Format("##.#", 26.56789) // Returned value = "26.6"
Tag	Format("#.##", 26.56789) // Returned value = "26.57"
Tag	Format("##.##", 26.56789) // Returned value = "26.57"

19.5.6 GetBit

"GetBit" is an internal scripting function that determines the value of a single bit in a numeric value.

Function	Group	Execution	Windows	Embedded	Thin Client
GetBit	Arithmetic	synchronous	supported	supported	supported

Syntax

```
GetBit( tagName, numBitNumber )
```

Inputs

tagName

The name of an Integer tag from which the bit value is called.



To directly specify the name of a tag instead of using the value of the tag, the tag name has to be enclosed in quotation marks.

For example, **GetBit("Second", 1)**.

numBitNumber

A numeric tag or value specifying the position (0...31) of the bit to call.

Returned value Returns the value ("0" or "1") of the specified bit.

Appendix: WinStudio-Internal Scripting Language



Individual bits in Integer tags can also be read and written by means of the "Bit field" (variable attribute).

For example, enter "Second->b0" to access the LSB (Least Significant Bit of the Second tag) and "Second->b31" to access the MSB (Most Significant Bit of the Second tag).

Examples

Tag name	Expression
Tag	GetBit("numSource", 4) // If the tag numSource holds a value of 15, then this function returns 0.
Tag	GetBit("numSource", 1) // If the tag numSource holds a value of 19, then this function returns 1.

19.5.7 Mod

"Mod" is an internal scripting function that returns the remainder from a division.

Function	Group	Execution	Windows	Embedded	Thin Client
Mod	Arithmetic	synchronous	supported	supported	supported

Syntax

```
Mod( numNumerator, numDenominator )
```

Inputs

numNumerator

Integer or Real tag containing the numerator of the function (dividend).

numDenominator

Integer or Real tag containing the denominator of the function (divisor).

Returned value

Returns the remainder (as a real number) after dividing "numNumerator" by "numDenominator".



Use the [Div, page 485](#) function to receive the whole-number dividend of the operation.

Examples

Tag name	Expression
Tag	Mod(50, 4) // Returned value = 2
Tag	Mod(16, 4) // Returned value = 0
Tag	Mod(100, 8.2) // Returned value = 1.600

19.5.8 Pow

Pow is an internal scripting function that calculates the power of a base.

Function	Group	Execution	Windows	Embedded	Thin Client
Pow	Arithmetic	synchronous	supported	supported	supported

Syntax

```
Pow( numBase, numExponent )
```

Appendix: WinStudio-Internal Scripting Language

Inputs	numBase						
	Integer or Real tag containing the value of the base of the function.						
	numExponent						
	Returns the result of the exponential function.						
Returned value	Returns the power of the base with the exponent.						
Examples	<table><tr><th>Tag name</th><th>Expression</th></tr><tr><td>Tag</td><td>Pow(2, 3) // Returned value = 8</td></tr><tr><td>Tag</td><td>Pow(10, 4) // Returned value = 10000</td></tr></table>	Tag name	Expression	Tag	Pow(2, 3) // Returned value = 8	Tag	Pow(10, 4) // Returned value = 10000
Tag name	Expression						
Tag	Pow(2, 3) // Returned value = 8						
Tag	Pow(10, 4) // Returned value = 10000						

19.5.9 ResetBit

Resets a single bit in an Integer tag to 0.

Function	Group	Execution	Windows	Embedded	Thin Client
ResetBit	Arithmetic	synchronous	supported	supported	supported

Syntax

```
ResetBit( tagName, numBitNumber )
```

Inputs

tagName

The name of an Integer tag in which the bit value is to be reset.

To directly specify the name of a tag instead of using the value of the tag, the tag name has to be enclosed in quotation marks.

For example, **ResetBit("Second", 1)**.

numBitNumber

A numeric tag or value specifying the position (0...31) of the bit to reset.

Returned value

Value	Description
0	No error
1	Invalid parameter
2	Tag does not exist

Individual bits in Integer tags can also be read and written by means of the "Bit field" (variable attribute).

For example, enter **Second->b0** to access the LSB (Least Significant Bit of the Second tag) and **Second->b31** to access the MSB (Most Significant Bit of the Second tag).

Examples

Appendix: WinStudio-Internal Scripting Language

Tag name	Expression
Tag	ResetBit("numSource", 4) // If the tag numSource held a value of 16, then the function returns 0 and numSource holds a new value of 0.
Tag	ResetBit("numSource", 1) // If the tag numSource held the value 19, then the function returns 0 and numSource holds a new value of 17.

19.5.10 Round

Rounds "numValue" to the nearest integer.

Function	Group	Execution	Windows	Embedded	Thin Client
Round	Arithmetic	synchronous	supported	supported	supported

Syntax

```
Round( numValue )
```

Inputs

numValue

A Real tag that holds the value to be rounded.

Returned value

Returns the integer result of the round function.

Examples

Tag name	Expression
Tag	Round("345.87") // Returned value = 346
Tag	Round("65.323") // Returned value = 65

19.5.11 SetBit

Sets a single bit in an Integer tag to 1.

Function	Group	Execution	Windows	Embedded	Thin Client
SetBit	Arithmetic	synchronous	supported	supported	supported

Syntax

```
SetBit( tagName, numBitNumber )
```

Inputs

tagName

The name of an Integer tag where the bit value is set.



To directly specify the name of a tag instead of using the value of the tag, the tag name has to be enclosed in quotation marks.

For example, **SetBit("Second", 1)**.

numBitNumber

A numeric tag or value specifying the position (0...31) of the bit to set.

Appendix: WinStudio-Internal Scripting Language

Returned value

Value	Description
0	No error
1	Invalid parameter
2	Tag does not exist



Individual bits in Integer tags can also be read and written by means of the "Bit field" (variable attribute).

For example, enter "Second->b0" to access the LSB (Least Significant Bit of the Second tag) and "Second->b31" to access the MSB (Most Significant Bit of the Second tag).

Examples

Tag name	Expression
Tag	SetBit("numSource", 4) // If the tag numSource held a value of 0, then this function returns 0 and numSource holds a new value of 16.
Tag	SetBit("numSource", 1) // If the tag numSource held the value 17, then this function returns 0 and numSource holds a new value of 19.

19.5.12 Sqrt

Calculates the square root of "numValue".

Function	Group	Execution	Windows	Embedded	Thin Client
Sqrt	Arithmetic	synchronous	supported	supported	supported

Syntax

Sqrt(numValue)

Inputs

numValue

Integer or Real tag to be square rooted.

Returned value

Returns the square root of the value in the "numValue" tag.



If "numValue" has a negative value, this function returns the value 0 and sets the quality of the returned tag to "BAD" (quality display in the Database Spy).

Examples

Tag name	Expression
Tag	SQRT(25) // Returns the value 5
Tag	SQRT(67) // Returns the value 8.185353

19.5.13 Swap16

Swaps the two lower bytes of a tag.

Function	Group	Execution	Windows	Embedded	Thin Client
Swap16	Arithmetic	synchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Syntax

```
Swap16( numValue )
```

Inputs

numValue

Integer tag that holds the numeric value of the bytes to be swapped.

Returned value

Returns the numeric value after swapping the bytes.

Examples

Tag name	Expression
Tag	Swap16(16) // 16 = 0000000000010000 in binary. Returned value = 4096 = 0001000000000000 in binary.
Tag	Swap16(43760) // 43760 = 1010010111110000 in binary. Returned value = 61610 = 1111000010100101 in binary.

19.5.14 Swap32

Swaps two data words of a tag.

Function	Group	Execution	Windows	Embedded	Thin Client
Swap32	Arithmetic	synchronous	supported	supported	supported

Syntax

```
Swap32( numValue )
```

Inputs

numValue

Integer tag that holds the numeric value of the data words to be swapped.

Returned value

Returns the numeric value after swapping the data words.

Examples

Tag name	Expression
Tag	Swap32(16) // 16 = 00000000000000000000000000000000 in binary. Returned value = 1048576 = 00000000000100000000000000000000 in binary.
Tag	Swap32(246333120) // 286333120 = 101010101010101010111111100000000 in binary. Returned value = -1094709586 = 11111111000000001010101010101010 in binary.

19.5.15 Trunc

Returns the whole-number portion of a real numeric value.

Function	Group	Execution	Windows	Embedded	Thin Client
Trunc	Arithmetic	synchronous	supported	supported	supported

Syntax

Trunc(numValue)

Inputs

numValue

Real tag of which the whole-number portion is to be calculated.

Appendix: WinStudio-Internal Scripting Language

Returned value Returns the whole-number portion of the real numeric value of "numValue", i.e., all decimal places are truncated without rounding.

Examples

Tag name	Expression
	Trunc(234.987) // Returned value = 234
	Trunc(-3465.9) // Returned value = -3465

19.6 Statistical Functions

19.6.1 Statistical Functions – General Information

These functions are used to calculate specific statistics – such as average, maximum, and minimum – from two or more numeric values.

Statistical

Function	Execution	Windows PC	Windows CE	Thin Client
Avg, page 495	synchronous	supported	supported	supported
Max, page 496	synchronous	supported	supported	supported
Min, page 497	synchronous	supported	supported	supported
Rand, page 498	synchronous	supported	supported	supported

19.6.2 Avg

Returns the average value of a set of numbers.

Function	Group	Execution	Windows	Embedded	Thin Client
Avg	Statistical	synchronous	supported	supported	supported

Syntax

```
Avg( numValue1, numValue2, ... , numValueN )
Avg( "tagArray", numSample, optNumIgnore )
```

This function has two formats:

- If the first parameter is a numeric tag or value, the **Avg(numValue1, numValue2, ... , numValueN)** format has to be used.
- If the first parameter is an array tag in quotation marks or a String tag, the **Avg("tagArray", numSample, optNumIgnore)** format has to be used.

Inputs

numValue (1...N)

Integer or Real tags containing the numbers to be averaged.

tagArray

Name of an array tag (Real or Integer) containing the values to be averaged.

numSample

The number of array elements to be averaged.

Appendix: WinStudio-Internal Scripting Language

optNumIgnore

Optional Integer or Real tag containing the value to be ignored when calculating the average.

Returned value Returns the average of the values.

Examples

Tag name	Expression
Tag	Avg(1, 2.34, 5, 7, 4, 8, 9.4) // Returned value = 5.248571
Tag	Avg(1, 5, -9, 0, 5, 3) // Returned value = 0.833333

Tag name	Expression
Tag	Avg("tagArray[1]", 3) // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 30
Tag	Avg("tagArray[1]", 3, 10) // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 40

19.6.3 Max

Returns the maximum value of a set of numbers.

Function	Group	Execution	Windows	Embedded	Thin Client
Max	Statistical	synchronous	supported	supported	supported

Syntax

```
Max( numValue1, numValue2, ... , numValueN )
Max( "tagArray", numSample, optNumIgnore )
```



This function has two formats:

- If the first parameter is a numeric tag or value, the **Max(numValue1, numValue2, ... , numValueN)** format is used.
- If the first parameter is an array tag in quotation marks or a string tag, the **Max("tagArray", numSample, optNumIgnore)** format is used.

Inputs**numValue (1...N)**

Integer or Real tags containing the numbers to be analyzed.

tagArray

Name of an array tag (Real or Integer) containing the values to be analyzed.

numSample

The number of array elements to be analyzed.

optNumIgnore

Integer or Real tags containing the value to be ignored in the analysis.

Returned value Returns the maximum value of the set.

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	Max(1, 2.34, 5, 7, 4, 8, 9.4) // Returned value = 9.4
Tag	Max(1, 5, -9, 0, 5, 3) // Returned value = 5
Tag	Max("tagArray[1]", 3) // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 60
Tag	Max("tagArray[1]", 3, 10) // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 60

19.6.4 Min

Returns the minimum value of a set of numbers.

Function	Group	Execution	Windows	Embedded	Thin Client
Min	Statistical	synchronous	supported	supported	supported

Syntax

```
Min( numValue1, numValue2, ... , numValueN )
Min( "tagArray", numSample, optNumIgnore )
```



This function has two formats:

- If the first parameter is a numeric tag or value, the **Min(numValue1, numValue2, ... , numValueN)** format is used.
- If the first parameter is an array tag in quotation marks or a String tag, the **Min("tagArray", numSample, optNumIgnore)** format is used.

Inputs

numValue (1...N)

Integer or Real tags containing the numbers to be analyzed.

tagArray

Name of an array tag (Real or Integer) containing the values to be analyzed.

numSample

The number of array elements to be analyzed.

optNumIgnore

Integer or Real tags containing a value to be ignored in the analysis.

Returned value

Returns the minimum value of the set.

Examples

Tag name	Expression
Tag	Min(1, 2.34, 5, 7, 4, 8, 9.4) // Returned value = 1
Tag	Min(1, 5, -9, 0, 5, 3) // Returned value = -9

Appendix: WinStudio-Internal Scripting Language

Tag name	Expression
Tag	Min("tagArray[1]", 3) // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 10
Tag	Min("tagArray[1]", 3, 10) // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 20

19.6.5 Rand

Generates a random number between 0 and 1.

Function	Group	Execution	Windows	Embedded	Thin Client
Border	Statistical	synchronous	supported	supported	supported

Syntax

Rand()

This function does not have any parameters.

Returned value

Returns a real number between 0 and 1.

Examples

Tag name	Expression
Tag	Rand() // Returned value = ?, Where: 0<?<1

19.7 Logarithmic Functions

19.7.1 Logarithmic Functions – General Information

These functions are used to perform logarithmic operations on numeric values.

Logarithmic

Function	Execution	Windows PC	Windows CE	Thin Client
Exp, page 498	synchronous	supported	supported	supported
Log, page 499	synchronous	supported	supported	supported
Log10, page 499	synchronous	supported	supported	supported

19.7.2 Exp

Calculates the value of Euler's number "e" (2.718282) raised to the power of "numValue".

Function	Group	Execution	Windows	Embedded	Thin Client
Exp	Logarithmic	synchronous	supported	supported	supported

Syntax

Exp(numValue)

Inputs

numValue

Integer or Real tag containing the value of the exponent of this function.

Returned value

Returns the value of "e^(numValue)".

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	Exp(1) // Returned value = 2.718282
Tag	Exp(5.25896) // Returned value = 192.281415

19.7.3 Log

Calculates the natural logarithm of "numValue".

Function	Group	Execution	Windows	Embedded	Thin Client
Log	Logarithmic	synchronous	supported	supported	supported

Syntax

Log(numValue)

Inputs

numValue

Integer or Real tag from which the natural logarithm is taken.

Returned value

Returns the value of "ln(numValue)".



If "numValue" has a negative value, this function returns the value 0 and sets the quality of the returned tag to "BAD" (quality display in the Database Spy).

Examples

Tag name	Expression
Tag	Log(2.718282) // Returned value = 1
Tag	Log(100) // Returned value = 4.605170

19.7.4 Log10

Calculates the logarithm "numValue" to the base of 10.

Function	Group	Execution	Windows	Embedded	Thin Client
Log10	Logarithmic	synchronous	supported	supported	supported

Syntax

Log10(numValue)

Inputs

numValue

Integer or Real tag, from which the logarithm to the base of 10 is taken.

Returned value

Returns the value of "ln(numValue)".



If "numValue" has a negative value, this function returns the value 0 and sets the quality of the returned tag to "BAD" (quality display in the Database Spy).

Examples

Tag name	Expression
Tag	Log10(1000) // Returned value = 3
Tag	Log10(43.05) // Returned value = 1.633973

Appendix: WinStudio-Internal Scripting Language

19.8 Logical functions

19.8.1 Logical functions – General Information

These functions are used to perform logical operations (e.g., if/then, true/false) on tags and expressions.

Logical

Function	Execution	Windows PC	Windows CE	Thin Client
False, page 501	synchronous	supported	supported	supported
If, page 501	synchronous	supported	supported	supported
Toggle, page 502	synchronous	supported	supported	supported
True, page 503	synchronous	supported	supported	supported

19.8.2 Logical Operators

The internal scripting language supports the following logic operators.

Logical operators

Operator	Usage	Description
AND	A AND B	TRUE if A and B are both TRUE
OR	A OR B	TRUE if A is TRUE, or B is TRUE, or both
XOR	A XOR B	TRUE if A is TRUE, or B is TRUE, but not both
NOT	NOT A	TRUE if A is FALSE
=	X = Y	TRUE if X is equal to Y
>	X > Y	TRUE if X is greater than Y
>=	X >= Y	TRUE if X is greater than or equal to Y
<	X < Y	TRUE if X is less than Y
<=	X <= Y	TRUE if X is less than or equal to Y
<>	X <> Y	TRUE if X is not equal to Y
&	X & Y	Bitwise AND: 0101 (decimal 5) AND 0011 (decimal 3) = 0001 (decimal 1)
	X Y	Bitwise OR: 0101 (decimal 5) OR 0011 (decimal 3) = 0111 (decimal 7)
^	X ^ Y	Bitwise XOR: 0101 (decimal 5) XOR 0011 (decimal 3) = 0110 (decimal 6)

Appendix: WinStudio-Internal Scripting Language

Operator	Usage	Description
~	~ X	Bitwise NOT: NOT 0101 (decimal 5) = 1010 (decimal 10)
>> n	X >> Y	Rotate n bits to right: 0110 (decimal 6) ROTATE RIGHT = 0011 (decimal 3)
<< n	X << Y	Rotate n bits to left: 0110 (decimal 6) ROTATE LEFT = 1100 (decimal 12)



For more complex logical operators, the [Logical, page 500](#) and [Loop, page 640](#) functions are to be used.

19.8.3 False

Determines whether the specified tag or expression is logically false.

Function	Group	Execution	Windows	Embedded	Thin Client
False	Logical	synchronous	supported	supported	supported

Syntax

False(TagOrExpression)

Inputs

TagOrExpression

Tag or expression to be used in the function.

Returned value

0	If the tag or expression is "not" logically false
1	If the tag or expression is logically false



This function is useful if an actual value of 0 needs to be returned in case the expression returns a value other than 0.

Examples

Tag name	Expression
Tag	False(1) // Returned value = 0
Tag	False(5 < 2) // Returned value = 1

19.8.4 If

Determines whether the contents of "numExpression" are logically true and returns the value of "numThen" or "optNumElse" accordingly.

Function	Group	Execution	Windows	Embedded	Thin Client
If	Statistical	synchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Syntax

```
If( numExpression, numThen, optNumElse )
```

Inputs**numExpression**

Tag or expression used as the condition in the function.

numThen

Tag or expression used if the condition is logically true.

optNumElse

Optional tag or expression used if the condition is logically false.

Returned value

numThen	If the numExpression is logically true
optNumElse	If the numExpression is logically false
No value returned	If the numExpression is logically false and there is no optNumElse in the function



The "numExpression" parameter can be a combination of logical operators ("AND", "OR" and "NOT").

For example, **If(TagA>TagB AND TagA=10,1,0)**.

The "numThen" parameter can be another function also using the "If()" function. For this reason "If()" functions can be nested.

For example, **if(TagA>TagB,If(TagA<TagC,1, 2),3)**.

Examples

Tag name	Expression
Tag	If(5 > 4,10, 6) // Returned value = 10
Tag	If(5 < 2, 0, 2) // Returned value = 0
Tag	If(3 = 9, 67) // No Returned value. (Tag retains previous value.)

19.8.5 Toggle

Returns the toggled value from the contents of "numValue" tag.

Function	Group	Execution	Windows	Embedded	Thin Client
Toggle	Logical	synchronous	supported	supported	supported

Syntax

```
Toggle( numValue )
```

Inputs**numValue**

Boolean tag containing the value to be toggled.

Returned value

Numeric result (0 or 1) of the value to be toggled.



This function does not actually change the value of the tag. However, the function can be used in an expression or command changing the value of the tag.

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	Toggle(MyBoolTag) // Returned value = 1 if MyBoolTag value equals 0, or 0 if MyBoolTag value equals 1
Tag	Toggle(numValue) // Returned value = toggled value of the number in the numValue tag

19.8.6 True

Determines whether the specified tag or expression is logically true.

Function	Group	Execution	Windows	Embedded	Thin Client
True	Logical	synchronous	supported	supported	supported

Syntax

True(TagOrExpression)

Inputs

TagOrExpression

Tag or expression to be used in the function.

Returned value

0	If the tag or expression is not logically true
1	If the tag or expression is logically true



This function is useful if an actual value of 1 needs to be returned in case the expression returns a value other than 0.

Examples

Tag name	Expression
Tag	True(1) // Returned value = 1
Tag	True(5 < 2) // Returned value = 0

19.9 String Functions

19.9.1 String Functions – General Information

These functions are used to manipulate text strings or convert them into numeric values.

Character string

Function	Execution	Windows PC	Windows CE	Thin Client
Asc2Str, page 504	synchronous	supported	supported	supported
CharToValue, page 505	synchronous	supported	supported	supported
CharToValueW, page 505	synchronous	supported	supported	supported
ClassMembersToStrVector, page 506	synchronous	supported	supported	supported
NCopy, page 507	synchronous	supported	supported	supported
Num, page 508	synchronous	supported	supported	supported
Str, page 508	synchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Function	Execution	Windows PC	Windows CE	Thin Client
Str2Asc, page 509	synchronous	supported	supported	supported
StrCompare, page 509	synchronous	supported	supported	supported
StrCompareNoCase, page 510	synchronous	supported	supported	supported
StrFromInt, page 510	synchronous	supported	supported	supported
StrFromReal, page 511	synchronous	supported	supported	supported
StrFromTime, page 512	synchronous	supported	supported	supported
StrGetElement, page 513	synchronous	supported	supported	supported
StrLeft, page 513	synchronous	supported	supported	supported
StrLen, page 514	synchronous	supported	supported	supported
StrLower, page 514	synchronous	supported	supported	supported
StrRChr, page 514	synchronous	supported	supported	supported
StrRight, page 515	synchronous	supported	supported	supported
StrSetElement, page 515	synchronous	supported	supported	supported
StrStr, page 516	synchronous	supported	supported	supported
StrStrPos, page 516	synchronous	supported	supported	supported
StrTrim, page 517	synchronous	supported	supported	supported
StrTrimAll, page 518	synchronous	supported	supported	supported
StrUpper, page 518	synchronous	supported	supported	supported
ValueToChar, page 519	synchronous	supported	supported	supported
ValueWToChar, page 519	synchronous	supported	supported	supported

19.9.2 Asc2Str

This function converts one or more Unicode character codes to a string.

Function	Group	Execution	Windows	Embedded	Thin Client
Asc2Str	Character string	synchronous	supported	supported	supported

Syntax

```
Asc2Str( numChar1, numChar2, ... , numCharN )
```

Inputs

numChar (1-N)

A Unicode character code (in decimal).

Returned value

Returns a string with the converted codes.



Although the name of the function indicates that it only supports ASCII characters, it can be used for the entire Unicode character set.

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	Asc2Str(65) // Returned value = "A"
Tag	Asc2Str(65, 66, 67) // Returned value = "ABC"
Tag	Asc2Str(Array[0], Array[1], Array[2]) // Returned value = "ABC"

19.9.3 CharToValue

This function converts a string to Unicode character codes and stores those values in an Integer array.

Function	Group	Execution	Windows	Embedded	Thin Client
CharToValue	Character string	synchronous	supported	supported	supported

Syntax

```
CharToValue("tagString", "tagArray")
```

Inputs

tagString

The name of the String tag whose value is to be converted.

tagArray

The name of the Integer array that is to receive the converted values. If no array is specified, the default value is 0.



The tag names for tagString and tagArray have to be enclosed in quotation marks as shown in the syntax diagram. Otherwise, the project tries to get the values of the specified tag.

Returned value

Returns the number of array elements used, which has to be equal to the number of characters in the string.

Examples

If StrTag = "ABC", then Array[0] = 65, Array[1] = 66, and Array[2] = 67:

```
CharToValue( "StrTag", "Array" )
```

If StrTag = "ABC", then Array[10] = 65, Array[11] = 66, and Array[12] = 67:

```
CharToValue( "StrTag", "Array[10]" )
```

19.9.4 CharToValueW

This function converts a string to Unicode character codes, combines two codes into a double-byte word and stores those values in an Integer array.

Function	Group	Execution	Windows	Embedded	Thin Client
CharToValueW	Character string	synchronous	supported	supported	supported

Syntax

```
CharToValueW("tagString", "tagArray")
```

Inputs

tagString

The name of the String tag whose value is to be converted.

Appendix: WinStudio-Internal Scripting Language



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

tagArray

The name of the Integer array that is to receive the converted values. If no array is specified, the default value is 0.



The variable names for tagString and tagArray have to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the values of the specified tag.

Returned value

Returns the number of array elements used, which is equal to "half" the number of characters in the string.



Because of how each two character codes are combined into single value, this function only supports Unicode character codes 0 through 255. For character codes greater than 255 or if double-byte words are not needed, the [CharToValue page 505](#) function is to be used.

Examples

If StrTag = "Studio", then Array[0] = 29779 ("St"), Array[1] = 25717 ("ud"), and Array[2] = 28521 ("io"):

CharToValue("StrTag", "Array")

If StrTag = "Studio", then Array[10] = 29779 ("St"), Array[11] = 25717 ("ud"), and Array[12] = 28521 ("io"):

CharToValue("StrTag", "Array[10]")

19.9.5 ClassMembersToStrVector

Transfers values from a class tag to an array tag.

Function	Group	Execution	Windows	Embedded	Thin Client
ClassMembersToStrVector	Character string	synchronous	supported	supported	supported

Syntax

```
ClassMembersToStrVector( "strClassTag" , numStartPos, numNumPos, "strArrayTag", optBooStartPosTarget )
```

Inputs**strClassTag**

String value containing the class tag name.

numStartPos

Start position (array index) of "strClassTag" .

numNumPos

Number of positions (array indexes) to be transferred from "strClassTag" .

Appendix: WinStudio-Internal Scripting Language

strArrayTag

String value containing the array tag that is to receive the values from "strClassTag" .

optBooStartPosTarget

Start position (array index) of "strArrayTag". If omitted, the default value 1 is used.

Returned value

-6	Array size of "strClassTag" is not big enough for "numStartPos"
-5	"strClassTag" is not a Class tag
-4	"strClassTag" is not found
-3	"strArrayTag" is not found
-2	Invalid data type of the parameters
-1	Invalid number of parameters
0	Transferred successfully



If "strClassTag" has more than one element, the value of each class element is transferred to "strArrayTag". Therefore, it is important to make sure that the array size of "strArrayTag" is big enough to receive all values from "strClassTag".

Examples

Tag name	Expression
Tag	ClassMembersToStrVector("Classtag", 5, 3, "Arraytag")
Tag	ClassMembersToStrVector("Classtag", 5, 3, "Arraytag", 0)
Tag	ClassMembersToStrVector (TagName, 0, 1, ArrayName)

19.9.6 NCopy

Copies a defined section of a larger string.

Function	Group	Execution	Windows	Embedded	Thin Client
NCopy	Character string	synchronous	supported	supported	supported

Syntax

NCopy(strSource, numStartChar, numQtdChar)

Inputs

strSource

The source string.

numStartChar

Integer tag containing a number corresponding to the first character being copied.

numQtdChar

The number of characters to be copied.

Appendix: WinStudio-Internal Scripting Language

Returned value Returns a string that is part of the source string (as defined by the function).

Examples

Tag name	Expression
Tag	NCopy("Studio version 7.0", 7, 7) // Returned value = "version"
Tag	NCopy("Technical Reference", 0, 9) // Returned value = "Technical"



The value 0 is assigned to the first character in the string.

19.9.7 Num

Converts a string into a floating point number.

Function	Group	Execution	Windows	Embedded	Thin Client
Num	Character string	synchronous	supported	supported	supported

Syntax

```
Num( strValue )
```

Inputs

strValue

The number of characters to be converted into the floating point number format.

Returned value Returns the number (formerly in a string format) as floating point number.

Examples

Tag name	Expression
Tag	Num("321654.987") // Returned value = 321654.987
Tag	Num("5.6589626246") // Returned value = 5.6589626246



The string for the floating point number cannot use characters other than the numbers (0..9) and a decimal point (.). Otherwise, the function returns the value 0.0.

19.9.8 Str

Converts a number into a string.

Function	Group	Execution	Windows	Embedded	Thin Client
Str	Character string	synchronous	supported	supported	supported

Syntax

```
Str( numValue )
```

Inputs

numValue

Integer or floating point number tag containing a number to be converted to a string.

Returned value Returns the string in a floating point number format.

Examples

Tag name	Expression
Tag	Str(321654.987) // Returned value = "321654.987"
Tag	Str(5.65896246) // Returned value = "5.658962"

19.9.9 Str2Asc

This function converts a character to its original Unicode character code.

Function	Group	Execution	Windows	Embedded	Thin Client
Str2Asc	Character string	synchronous	supported	supported	supported

Syntax

Str2Asc(strChar)

Inputs

strChar

The character to be converted.

Returned value

Returns the Unicode character code (in decimal) for the specified character.



Although the name of the function indicates that it only supports ASCII characters, it can be used for the entire Unicode character set.

Examples

Tag name	Expression
Tag	Str2Asc("C") // Returned value = 67
Tag	Str2Asc("o") // Returned value = 111

19.9.10 StrCompare

Compares two strings to check if they are identical.

Function	Group	Execution	Windows	Embedded	Thin Client
StrCompare	Character string	synchronous	supported	supported	supported

Syntax

StrCompare(strValue1, strValue2)

Inputs

strValue1

A string or a tag of type [String on page 113](#). This is the first string to be compared.

strValue2

A string or a tag of type [String on page 113](#). This is the second string to be compared.

Returned value

-1	The value of ""strValue1"" is smaller than the value of ""strValue2"".
0	""strValue1"" and ""strValue2"" are identical.
1	The value of ""strValue1"" is greater than the value of ""strValue2"".

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	StrCompare("Text1", "Text2") // Returned value = -1
Tag1="Text" Tag2="Text"	StrCompare(Tag1, Tag2) // Returned value = 0
Tag1="Text1" Tag2="Text2"	StrCompare(Tag1, Tag2) // Returned value = -1

19.9.11 StrCompareNoCase

Compares two strings to check if they are identical. The case of characters is ignored (i.e., a lower-case "a" has the same value as an upper-case "A").

Function	Group	Execution	Windows	Embedded	Thin Client
StrCompareNo-Case	Character string	synchronous	supported	supported	supported

Syntax

```
StrCompareNoCase( strValue1, strValue2 )
```

Inputs

strValue1

A string or a tag of type [String on page 113](#). This is the first string to be compared.

strValue2

A string or a tag of type [String on page 113](#). This is the second string to be compared.

Returned value

-1	The value of strValue1 is less than the value of strValue2.
0	strValue1 and strValue2 are identical
1	The value of strValue1 is greater than the value of strValue2

Examples

Tag name	Expression
Tag	StrCompareNoCase("Text1", "TEXT1") // Returned value = "0"
Tag1="Text1" Tag2="TEXT1"	StrCompareNoCase(Tag1, Tag2) // Returned value = "0"

19.9.12 StrFromInt

Converts an integer into its respective string in another base number system, such as binary (base 2) or octal (base 8).

Function	Group	Execution	Windows	Embedded	Thin Client
StrFromInt	Character string	synchronous	supported	supported	supported

Syntax

```
StrFromInt( numValue, numBase )
```

Appendix: WinStudio-Internal Scripting Language

Inputs

numValue

The numeric value to be converted into a string.

numBase

The base number system to convert into.

Returned value

This function returns a string of the given integer in the specified base number system. The returned value can be stored in any String-type tag.

A real number can be specified instead of an integer. However, only the whole part of the number is converted. To convert the entire real number, use the [StrFromReal](#), [page 511](#) function instead.

If the base does not need to be changed, use the [Str](#), [page 508](#) function instead.

Examples

Tag name	Expression
Tag	"StrFromInt(26, 2)" // Returned value = ""11010""
Tag	"StrFromInt(26, 8)" // Returned value = ""32""

19.9.13 StrFromReal

Converts a real numeric value to a string, either as floating point number or in exponential notation.

Function	Group	Execution	Windows	Embedded	Thin Client
StrFromReal	Character string	synchronous	supported	supported	supported

Syntax

```
StrFromReal(numValue,numPrecision,{ strType | f | e | E })
```

Inputs

numValue

The numeric value to be converted.

numPrecision

The number of decimal places to be shown in the resulting string. Please note that the value is rounded instead of truncated.

strType

A single-character code that specifies how the resulting string is to be formatted, as described in the following table:

Value of strType	Description
f	Formatted in floating-point notation
e	Formatted in exponential notation with a lower-case "e"
E	Formatted in exponential notation with an upper-case "E"

Appendix: WinStudio-Internal Scripting Language

Returned value This function returns a string representation of the given numeric value with the specified precision and notation.

Examples StrFromReal(263.355, 2, "f")
 ...returns a string value of "263.36".
 StrFromReal(263.355, 2, "e")
 ...returns a string value of "2.63e+002".
 StrFromReal(263.355, 2, "E")
 ...returns a string value of "2.63E+002".

19.9.14 StrFromTime

Converts a time stamp from UTC standard notation into a formatted string, adjusted to reflect the Time Zone setting in the Control Panel of the local computer.

Function	Group	Execution	Windows	Embedded	Thin Client
StrFromTime	Character string	synchronous	supported	supported	supported

Syntax

```
StrFromTime( numUTCtime, numType )
```

Inputs

numUTCtime

An integer or a tag of type [Integer, page 113](#). A time stamp in UTC standard notation.

numType

An integer or a tag of type [Integer, page 113](#). Specifies the format of the resulting string, as described in the following table:

Value of numType	Description
1	Displays the "date" in the same format that is selected in the Control Panel on the local computer.
2	Displays the "time" in the same format selected in the Control Panel of the local computer
3	Displays a standard 24-character string that shows both date and time
4	Displays the abbreviated name of the day of the week
5	Displays the full name of the day of the week

Returned value This function returns a string representation of the given time stamp with the specified formatting. The returned value can be stored in any tag of type [String page 113](#).



The Coordinated Universal Time (UTC) standard counts the number of seconds elapsed since 12:00 AM GMT on January 1, 1970. Each day consists of 86,400 seconds.

Examples The examples below refer to a computer set to Eastern Standard Time (or UTC -05:00).

Appendix: WinStudio-Internal Scripting Language

Tag name	Expression
Tag	StrFromTime(86400, 1) // Returned value = "1/1/70"
Tag	StrFromTime(86400, 2) // Returned value = "07:00:00 PM"
Tag	StrFromTime(86400, 3) // Returned value = "Thu Jan 01 19:00:00 1970"
Tag	StrFromTime(86400, 4) // Returned value = "Thu"
Tag	StrFromTime(86400, 5) // Returned value = "Thursday"

19.9.15 StrGetElement

Gets a specific element from a string.

Function	Group	Execution	Windows	Embedded	Thin Client
StrGetElement	Character string	synchronous	supported	supported	supported

Syntax

```
StrGetElement( strSource, strDelimiter, numElementNumber )
```

Inputs

strSource

The source string.

strDelimiter

Character used as delimiter between the elements.

numElementNumber

Number of the element returned by the function. The first element has the number 1. The second element has the number 2 and so forth.

Returned value

Returns the element (string value) retrieved from "strSource".

Examples

Tag name	Expression
Tag	StrGetElement("a b c", " ", 2) // returned value = "b"
Tag	StrGetElement("a,b,c", ",", 3) // returned value = "c"

19.9.16 StrLeft

Copies the first characters of a larger string.

Function	Group	Execution	Windows	Embedded	Thin Client
StrLeft	Character string	synchronous	supported	supported	supported

Syntax

```
StrLeft( strSource, numQtdChar )
```

Inputs

strSource

The source string.

Appendix: WinStudio-Internal Scripting Language

numQtdChar

The number of characters to be copied.

Returned value Returns a string containing the left-most characters in the source string.

Examples

Tag name	Expression
Tag	StrLeft("Studio version 7.0", 8) // Returned value = Studio v
Tag	StrLeft ("Technical Reference", 9) // Returned value = Technical

19.9.17 StrLen

Determines the length of a string.

Function	Group	Execution	Windows	Embedded	Thin Client
StrLen	Character string	synchronous	supported	supported	supported

Syntax

StrLen(strSource)

Inputs

strSource

The string.

Returned value Returns an integer, i.e., the number of characters in the string.

Examples

Tag name	Expression
Tag	StrLen("Studio version 7.0") // Returned value = 18
Tag	StrLen("Technical Reference") // Returned value = 19

19.9.18 StrLower

Converts a string to all lower case characters.

Function	Group	Execution	Windows	Embedded	Thin Client
StrLower	Character string	synchronous	supported	supported	supported

Syntax

StrLower(strSource)

Inputs

strSource

The string to be converted.

Returned value Returns the string, where all the characters are in lower case.

Examples

Tag name	Expression
Tag	StrLower("Studio version 7.0") // Returned value = "studio version 7.0"
Tag	StrLower("Technical Reference") // Returned value = "technical reference"

19.9.19 StrRChr

Isolates the final occurrence of a character sequence within a string.

Appendix: WinStudio-Internal Scripting Language

Function	Group	Execution	Windows	Embedded	Thin Client
StrRChr	Character string	synchronous	supported	supported	supported

Syntax

StrRChr(strSource, strChrSequence)

Inputs

strSource

The source string.

strCharSequence

The reference string.

Returned value Returns a string following the last occurrence of a character within the string.

Examples

Tag name	Expression
Tag	StrRChr("Studio version 7.0", "i") // Returned value = "ion 7.0"
Tag	StrRChr("Technical Reference", "n") // Returned value = "nce"

19.9.20 StrRight

Copies the last characters in a larger string.

Function	Group	Execution	Windows	Embedded	Thin Client
StrRight	Character string	synchronous	supported	supported	supported

Syntax

StrRight(strSource, numQtdChar)

Inputs

strSource

The source string.

numQtdChar

The number of characters to be copied.

Returned value Returns a string containing the right-most characters in a source string.

Examples

Tag name	Expression
Tag	StrRight("Studio version 7.0", 8) // Returned value = "sion 7.0"
Tag	StrRight("Technical Reference", 9) // Returned value = "Reference"

19.9.21 StrSetElement

Sets a specific element in a string.

Function	Group	Execution	Windows	Embedded	Thin Client
StrSetElement	Character string	synchronous	supported	supported	supported

Syntax

StrSetElement(strSource, strDelimiter, numElementNumber, strValue)

Appendix: WinStudio-Internal Scripting Language

Inputs

strSource

The source string.

strDelimiter

Character used as delimiter between the elements.

numElementNumberNumber of the element where the string value is to be written by the function.
Numbering starts at index 1.**strValue**

String value to be written to the numElementNumber of the strSource String tag.

Returned value

Returns the string value updated with the "strValue".

Examples

Tag name	Expression
StringTag	StrSetElement(strSource, " ", 2, "abcd")
StringTag	StrSetElement(strSource, ",", 3, "defg")

19.9.22 StrStr

Isolates the first occurrence of a character sequence within a string.

Function	Group	Execution	Windows	Embedded	Thin Client
StrStr	Character string	synchronous	supported	supported	supported

Syntax

StrStr(strSource, strCharSequence)

Inputs

strSource

The source string.

strCharSequence

The reference string.

Returned value

Returns the string of characters following the first occurrence of a character within the source string.

Examples

Tag name	Expression
Tag	StrStr("Studio version 7.0", "i") // Returned value = "io version 7.0"
Tag	StrStr("Technical Reference", "n") // Returned value = "nical Reference"

19.9.23 StrStrPos

Finds the first occurrence of a character within a string.

Appendix: WinStudio-Internal Scripting Language

Function	Group	Execution	Windows	Embedded	Thin Client
StrStrPos	Character string	synchronous	supported	supported	supported

Syntax

StrStrPos(strSource, strCharSequence)

Inputs

strSource

The source string.

strCharSequence

The reference string.

Returned value

Returns an integer corresponding to the first occurrence of a character within the string.

Examples

Tag name	Expression
Tag	StrStrPos("Studio version 7.0", "i") // Returned value = 4
Tag	StrStrPos("Technical Reference", "a") // Returned value = 7



The value 0 is assigned to the first character in the string.

19.9.24 StrTrim

Removes spaces from a string.

Function	Group	Execution	Windows	Embedded	Thin Client
StrTrim	Character string	synchronous	supported	supported	supported

Syntax

StrTrim(strReference, optNumFlag)

Inputs

strReference

A string or tag of type [String](#), [page 113](#) containing the source string.

optNumFlag

An "optional" integer or tag of type [Integer](#), [page 113](#):

Value of optNum-Flag	Description
0	Removes all spaces from both the beginning and the end of the string
1	Removes all spaces only from the beginning of the string
2	Removes all spaces only from the end of the string
3	Removes all spaces except for single spaces between words



If no value is specified for "optNumFlag", then "0" is the default.

Appendix: WinStudio-Internal Scripting Language

Returned value This function returns a string equal to "strReference" minus the specified space characters. The returned value can be stored in any tag of type [String](#) [page 113](#).

Examples

Tag name	Expression
Tag	StrTrim(" Studio version 7.0 ") // Returned value = "Studio version 7.0"
Tag	StrTrim(" Studio version 7.0 ", 0) // Returned value = "Studio version 7.0"
Tag	StrTrim(" Studio version 7.0 ", 1) // Returned value = "Studio version 7.0"
Tag	StrTrim(" Studio version 7.0 ", 2) // Returned value = "Studio version 7.0"
Tag	StrTrim(" Studio version 7.0 ", 3) // Returned value = "Studio version 7.0"

19.9.25 StrTrimAll

Eliminates a specific character from the whole string.

Function	Group	Execution	Windows	Embedded	Thin Client
StrTrimAll	Character string	synchronous	supported	supported	supported

Syntax

```
StrTrimAll( strReference, optStrTrimChar )
```

Inputs**strReference**

The source string.

optStrTrimChar

Character that is to be removed from the string. If this parameter is omitted, the space character is removed from the string by default.

Returned value Returns a string similar to "strReference" minus the characters removed by the function.

Examples

Tag name	Expression
Tag	StrTrimAll("Studio version 7.0 ", " ") // Returned value = "Studioversion7.0"

19.9.26 StrUpper

Converts a string to all upper case characters.

Function	Group	Execution	Windows	Embedded	Thin Client
StrUpper	Character string	synchronous	supported	supported	supported

Syntax

```
StrUpper( strSource )
```

Inputs**strSource**

The string.

Appendix: WinStudio-Internal Scripting Language

Returned value Returns the string with all characters in uppercase.

Examples

Tag name	Expression
Tag	StrUpper("Studio version 7.0") // Returned value = "STUDIO VER-SION 7.0"
Tag	StrUpper("Technical Reference") // Returned value = "TECHNICAL REFERENCE"

19.9.27 ValueToChar

This function converts an integer array of Unicode characters to a string.

Function	Group	Execution	Windows	Embedded	Thin Client
ValueToChar	Character string	synchronous	supported	supported	supported

Syntax

```
ValueToChar( "tagArray", numQtdChars )
```

Inputs

tagArray

The name of the integer array that is to receive the converted values. If no array index is specified, the default is 0. The tag name has to be enclosed in quotation marks, as shown in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

numQtdChars

The number of values to be converted (minimum of 1), starting with the specified array index.

Returned value Returns a string with the converted values.

Examples

If Array[0] = 65, Array[1] = 66 and Array[2] = 67, the returned value is "ABC":

```
ValueToChar( "Array", 3 )
```

If Array[10] = 65, Array[11] = 66 and Array[12] = 67, the returned value is "ABC":

```
ValueToChar( "Array[10]", 3 )
```

19.9.28 ValueWToChar

This function converts an integer array of Unicode characters to a string where each value in the array is a double-byte word.

Function	Group	Execution	Windows	Embedded	Thin Client
ValueWToChar	Character string	synchronous	supported	supported	supported

Syntax

```
ValueWToChar( "tagArray", numQtdChars )
```

Inputs

tagArray

The name of the integer array containing the double-byte values to be converted. If no array index is specified, the default value is 0. The tag name has to be enclosed in quotation marks, as shown in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Appendix: WinStudio-Internal Scripting Language

numQtdChars

The number of values to be converted (minimum of 1), starting with the specified array index.

Returned value Returns a string with the converted values.



Each value in the array has to be a double-byte word and each word is split into two bytes for conversion. As such, this function only supports Unicode characters 0 through 255. For characters greater than 255 or if double-byte words are not needed, use the [ValueToChar page 519](#) function.

Examples If Array[0] = 29779, Array[1] = 25717 and Array[2] = 28521, the returned value is "Studio":

ValueWToChar("Array", 3)

If Array[10] = 29779, Array[11] = 25717 and Array[12] = 28521, the returned value is "Studio":

ValueWToChar("Array[10]", 3)

19.10 Date and Time Functions

19.10.1 Date & Time functions – General Information

These functions are used to interact with the system clock or manipulate time stamps.

Date & Time

Function	Execution	Windows PC	Windows CE	Thin Client
ClockGetDate, page 520	synchronous	supported	supported	supported
ClockGetDayOfWeek, page 521	synchronous	supported	supported	supported
ClockGetTime, page 522	synchronous	supported	supported	supported
DateTime2Clock, page 522	synchronous	supported	supported	supported
GetClock, page 523	synchronous	supported	supported	supported
Hour2Clock, page 523	synchronous	supported	supported	supported
SetSystemDate, page 524	synchronous	supported	supported	supported
SetSystemTime, page 524	synchronous	supported	supported	supported

19.10.2 ClockGetDate

Calculates the date, based on how many seconds have elapsed since 12:00 AM GMT on January 1, 1970 (taking into account the current time zone of the computer).

Function	Group	Execution	Windows	Embedded	Thin Client
ClockGetDate	Date & Time	synchronous	supported	supported	supported

Syntax

ClockGetDate(numSeconds)

Appendix: WinStudio-Internal Scripting Language

Inputs

numSeconds

The number of seconds elapsed since 12:00 AM GMT on January 1, 1970.

Returned value

Returns the date calculated in String format.

Examples

Tag name	Expression
Tag	ClockGetDate(0) // If the computer is in the Central time zone. Returned value = 12/31/1969
Tag	ClockGetDate(1018886359) // If the computer is in the Central time zone. Returned value = 04/15/2002



This function takes the current time zone into account, as specified in the Control Panel of the local computer.

19.10.3 ClockGetDayOfWeek

Calculates the day of the week based on how many seconds have elapsed since 12:00 AM GMT on January 1, 1970 (taking into account the current time zone of the local computer).

Function	Group	Execution	Windows	Embedded	Thin Client
ClockGetDayOf-Week	Date & Time	synchronous	supported	supported	supported

Syntax

ClockGetDayOfWeek(numSeconds)

Inputs

numSeconds

The number of seconds elapsed since 12:00 AM GMT on January 1, 1970

Returned value

Returns the day of the week (calculated in integer format) as follows:

0	Sunday
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday

Examples

Tag name	Expression
Tag	ClockGetDayOfWeek(0) // If the computer is in the Central time zone. Returned value = 3
Tag	ClockGetDayOfWeek(1018886359) // If the computer is in the Central time zone. Returned value = 1



This function takes the current time zone into account, as specified in the Control Panel of the local computer.

Appendix: WinStudio-Internal Scripting Language

19.10.4 ClockGetTime

Calculates the time based on how many seconds have elapsed since 12:00 AM GMT on January 1, 1970 (taking into account the current time zone as specified on the local computer).

Function	Group	Execution	Windows	Embedded	Thin Client
ClockGetTime	Date & Time	synchronous	supported	supported	supported

Syntax

```
ClockGetTime( numSeconds )
```

Inputs

numSeconds

The number of seconds elapsed since 12:00 AM GMT on January 1, 1970.

Returned value

Returns the time calculated in String format.

Examples

Tag name	Expression
Tag	ClockGetTime(0) // If the computer is in the Central time zone. Returned value = 18:00:00
Tag	ClockGetTime(1018886359) // If the computer is in the Central time zone. Returned value = 10:59:19



This function takes the current time zone into account, as specified in the Control Panel of the local computer.



To convert the number of seconds strictly into the "HH:MM:SS" format, the [Format\(\)](#), [page 486](#) function has to be used instead of the "ClockGetTime()" function.

19.10.5 DateTime2Clock

Calculates how many seconds have elapsed since 12:00 AM GMT on January 1, 1970 (taking into account the current time zone specified on the local computer.)

Function	Group	Execution	Windows	Embedded	Thin Client
DateTime2Clock	Date & Time	synchronous	supported	supported	supported

Syntax

```
DateTime2Clock( strDate, strTime )
```

Inputs

strDate

The date to be used in the calculation.

strTime

The time to be used in the calculation.

Returned value

Returns the number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970.

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	DateTime2Clock("12/31/1969", "18:00:00") // If the computer is in the Central time zone. Returned value = 0
Tag	DateTime2Clock("04/15/2002", "10:59:19") // If the computer is in the Central time zone. Returned value = 01018886359



This function takes the current time zone into account, as specified in the Control Panel of the local computer.

19.10.6 GetClock

Calculates how many seconds have elapsed since 12:00 AM GMT on January 1, 1970, at the moment the function was run (taking into account the current time zone, as specified on the local computer).

Function	Group	Execution	Windows	Embedded	Thin Client
GetClock	Date & Time	synchronous	supported	supported	supported

Syntax

GetClock()

This function does not have any parameters.

Returned value

Returns the number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970, at the moment the function was run.

Examples

Tag name	Expression
Tag	GetClock() // If executed at 10:59:19 AM April 15th 2002 CST. Returned value = 101886359
Tag	GetClock() // If executed at 00:00:00 January 1st 1970 GMT. Returned value = 0



This function takes the current time zone into account, as specified in the Control Panel of the local computer.

19.10.7 Hour2Clock

Converts time in the "HH:MM:SS" format into seconds.

Function	Group	Execution	Windows	Embedded	Thin Client
Hour2Clock	Date & Time	synchronous	supported	supported	supported

Syntax

Hour2Clock(strTime)

Inputs

strTime

The number of hours, minutes and seconds in "HH:MM:SS" format.

Returned value

Returns the number of seconds corresponding to the total number of hours, minutes and seconds specified.

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	Hour2Clock("01:00:00") // Returned value = 3600
Tag	Hour2Clock("10:01:01") // Returned value = 36061

19.10.8 SetSystemDate

Sets the date in the clock of the operating system.

Function	Group	Execution	Windows	Embedded	Thin Client
SetSystemDate	Date & Time	synchronous	supported	supported	supported

Syntax

SetSystemDate(strDate)

Inputs

strDate

The date in "MM/DD/YYYY" format to set the clock.

Returned value

No returned values.

Examples

Tag name	Expression
	SetSystemDate("04/15/2002") // Sets the system clock to April 15th 2002.

19.10.9 SetSystemTime

Sets the time in the clock of the operating system.

Function	Group	Execution	Windows	Embedded	Thin Client
SetSystemTime	Date & Time	synchronous	supported	supported	supported

Syntax

SetSystemTime(strTime)

Inputs

strTime

The time in "HH:MM:SS" format to set the clock.

Returned value

No returned values.

Examples

Tag name	Expression
	SetSystemTime("15:45:18") // Sets the system clock to 3:45:18 PM.

19.11 Trigonometric Functions

19.11.1 Trigonometric Functions – General Information

These functions are used to calculate trigonometric values (e.g., sine, cosine, tangent).

Appendix: WinStudio-Internal Scripting Language

Trigonometric

Function	Execution	Windows PC	Windows CE	Thin Client
ACos, page 525	synchronous	supported	supported	supported
ASin, page 525	synchronous	supported	supported	supported
ATan, page 526	synchronous	supported	supported	supported
Cos, page 526	synchronous	supported	supported	supported
Cot, page 526	synchronous	supported	supported	supported
Pi, page 527	synchronous	supported	supported	supported
Sin, page 527	synchronous	supported	supported	supported
Tan, page 527	synchronous	supported	supported	supported

19.11.2 ACos

Calculates the Arc Cosine of a value.

Function	Group	Execution	Windows	Embedded	Thin Client
ACos	Trigonometric	synchronous	supported	supported	supported

Syntax

```
ACos( numValue )
```

Inputs

numValue

Numeric tag for the value of which the Arc Cosine is calculated.

Returned value

Returns the Arc Cosine of "numValue" in radians.

Examples

Tag name	Expression
Tag	ACos(1) // Returned value = 0.000000
Tag	ACos(0) // Returned value = 1.570796

19.11.3 ASin

Calculates the Arc Sine of a value.

Function	Group	Execution	Windows	Embedded	Thin Client
ASin	Trigonometric	synchronous	supported	supported	supported

Syntax

```
ASin( numValue )
```

Inputs

numValue

Numeric tag for the value of which the Arc Sine is calculated.

Returned value

Returns the Arc Sine of "numValue" in radians.

Examples

Tag name	Expression
Tag	ASin(1) // Returned value = 1.570796
Tag	ASin(0) // Returned value = 0.000000

Appendix: WinStudio-Internal Scripting Language

19.11.4 ATan

Calculates the Arc Tangent of a value.

Function	Group	Execution	Windows	Embedded	Thin Client
ATan	Trigonometric	synchronous	supported	supported	supported

Syntax

ATan(numValue)

Inputs**numValue**

Numeric tag for the value of which the Arc Tangent is calculated.

Returned value

Returns the Arc Tangent of "numValue" in radians.

Examples

Tag name	Expression
Tag	ATan(1) // Returned value = 0.785398
Tag	ATan(0) // Returned value = 1.570796

19.11.5 Cos

Calculates the Cosine of a value.

Function	Group	Execution	Windows	Embedded	Thin Client
Cos	Trigonometric	synchronous	supported	supported	supported

Syntax

Cos(numAngle)

Inputs**numAngle**

The angle (in radians) the Cosine of which is calculated.

Returned value

Returns the Cosine of "numAngle".

Examples

Tag name	Expression
Tag	Cos(1.570796) // Returned value = 0.000000
Tag	Cos(0) // Returned value = 1.000000

19.11.6 Cot

Calculates the Cotangent of a value.

Function	Group	Execution	Windows	Embedded	Thin Client
Cot	Trigonometric	synchronous	supported	supported	supported

Syntax

Cot(numAngle)

Inputs**numAngle**

The angle (in radians) the Cotangent of which is calculated.

Returned value

Returns the Cotangent of "numAngle".

Appendix: WinStudio-Internal Scripting Language

Although the Cotangent of 0 is mathematically infinite, "WinStudio" only returns the largest number possible.

Examples

Tag name	Expression
Tag	Cot(0.785398) // Returned value = 1.000000
Tag	Cot(0) // Returned value = 0.000000

19.11.7 Pi

Calculates the value of "pi".

Function	Group	Execution	Windows	Embedded	Thin Client
Pi	Trigonometric	synchronous	supported	supported	supported

Syntax

```
Pi()
```

This function does not have any parameters. However, the parentheses are still to be included as the function is otherwise recognized as tag instead as a function.

Returned value

Returns the value of "Pi".

Examples

Tag name	Expression
Tag	Pi() // Returned value = 3.141593

19.11.8 Sin

Calculates the Sine of a value.

Function	Group	Execution	Windows	Embedded	Thin Client
Sin	Trigonometric	synchronous	supported	supported	supported

Syntax

```
Sin( numAngle )
```

Inputs

numAngle

The angle (in radians) the Sine of which is calculated.

Returned value

Returns the Sine of "numAngle".

Examples

Tag name	Expression
Tag	Sin(0) // Returned value = 0.000000
Tag	Sin(1.570796) // Returned value = 1.000000

19.11.9 Tan

Calculates the Tangent of a value.

Function	Group	Execution	Windows	Embedded	Thin Client
Tan	Trigonometric	synchronous	supported	supported	supported

Syntax

```
Tan( numAngle )
```

Appendix: WinStudio-Internal Scripting Language

Inputs

numAngle

The angle (in radians) the Tangent of which is calculated.

Returned value

Returns the Tangent of "numAngle".

Although the Tangent of $\text{Pi}()$, [page 527](#) is mathematically infinite, "WinStudio" only returns the largest number possible.

Examples

Tag name	Expression
Tag	Tan(0) // Returned value = 0.00000
Tag	Tan(0.785398) // Returned value = 1.00000

19.12 Screen Functions

19.12.1 Screen Functions – General Information

These functions are used to open and close WinStudio screens.

Opening and Closing Windows

Function	Execution	Windows PC	Windows CE	Thin Client
Close, page 528	asynchronous	supported	supported	supported
Open, page 529	asynchronous	supported	supported	supported
OpenPrevious, page 531	asynchronous	supported	supported	supported
ShowInplaceInput, page 532	asynchronous	supported	supported	supported
ShowMessageBox, page 533	synchronous	supported	supported	supported

19.12.2 Close

"Close" is an internal scripting function that closes an open WinStudio screen.

Function	Group	Execution	Windows	Embedded	Thin Client
Close	Screens	asynchronous	supported	supported	supported

Syntax

```
Close(strScreen{ | ,optNumID })
```

Inputs

strScreen

The name of the screen (not including the ".scr" extension) to be closed.

Some web servers are case-sensitive. If the project is to be deployed as a web application, only lowercase letters are to be used for the screen name.

Appendix: WinStudio-Internal Scripting Language

optNumID

The specific ID or instance number of the screen to be closed, if there is more than one open screen with the same name. (The ID is assigned when the screen is opened with the [Open, page 529](#) function.)

This parameter is optional. If no value is specified, the default ID is 0.

Returned value This function does not return any value.



This function cannot be used in combination with [tasks, page 57](#) or in the [global procedures, page 716](#) script.

Additionally, this function does not need to be called to close a screen if another screen is opened, as the previous screen is automatically closed. For further information, refer to [Screen attributes, page 156](#).

Examples

Closing the WinStudio screen named "main":

Close("main")

Closing the WinStudio screen named "alarms":

Close("alarms")

Close the WinStudio screen named "main" with ID 10:

Close("main", 10)

19.12.3 Open

"Open" is an internal scripting function that opens a WinStudio screen.

Function	Group	Execution	Windows	Embedded	Thin Client
Open	Screens	asynchronous	supported	supported	supported

Syntax

```
Open(strScreen{ | ,optNumX1{ | ,optNumY1,optNumX2,optNumY2,optNumResizeFlag{ | ,optNumID{ |
,optStrMnemonicList } } } })
```

Inputs

strScreen

The name of the screen (not including the ".scr" extension) to be opened.



Some web servers are case-sensitive. If the project is to be deployed as a web application, only lowercase letters are to be used for the screen name.

Appendix: WinStudio-Internal Scripting Language

optNumX1, optNumY1, optNumX2, optNumY2

The coordinates in pixels for the upper-left (X1,Y1) and lower-right (X2,Y2) corners of the screen.

These are optional parameters. If no values are specified, the default screen size and location are used. For further information, refer to [Screen attributes, page 156](#).

Please note the following special cases:

- The screen can be opened at the current position of the cursor by using **Open("screen", 1)** or **Open("screen", 1, -1, -1, -1, ...)** if the parameters at the end are required.
- If **optNumX1** equals **optNumX2** and **optNumY1** equals **optNumY2**, the default screen size is used, however, the upper-left corner of the screen is positioned at (X1,Y1).
- If **optNumX2** is less than **optNumX1** and/or **optNumY2** is less than **optNumY1**, or if all four parameters are set to -1, the parameters are ignored and the default screen size and location are used.

optNumResizeFlag

Specifies whether the size of the screen objects is adjusted when opening the screen:

Value	Description
0	The size of screen objects is not adjusted.
1	Screen objects are automatically adjusted to fit the new dimensions of the screen, as specified by the coordinates described above. The adjustment is made when opening the screen. If the screen size is changed after opening, the objects are not readjusted.

This parameter is required if all four coordinates are specified.

optNumID

An ID or instance number is assigned to the screen as multiple instances of the same screen file can be opened. (This ID is required if a screen is closed using the [Close, page 528](#) function.)

This parameter is optional. If the value is missing, the default value of 0 is used.

optStrMnemonicList

A string that describes how the custom properties of generic objects or [linked symbols, page 200](#) in the screen are completed if the screen is opened. This string has the following syntax:

Label: Value

"Label" is the name of the property. "Value" is the tag, expression or direct value assigned to the property. More than one mnemonic can be declared as long as the mnemonics are separated by spaces. For further information, refer to "Examples" below.

Returned value This function returns one of the following values:

Appendix: WinStudio-Internal Scripting Language

Value	Description
0	Successful
1	Error



This function cannot be used in combination with [tasks, page 57](#) or in the [global procedures, page 716](#) script.

Examples

Opening the screen "main" using the default screen size and location:

Open("main")

Opening the screen at the current position of the cursor:

Open("main", 1)

Opening the screen at the current position of the cursor and assigning it an ID of 10:

Open("main", 1, -1, -1, -1, 0, 10)

Opening the screen using the default screen size, however, positioned at the coordinates (500,250) with the upper-left corner and assigning it an ID of 10:

Open("main", 500, 250, 500, 250, 0, 10)

Opening the screen using the default screen size and location and replacing the custom properties "Mne1" and "Mne2" by "Tag1" and "Tag2":

Open("main", -1, -1, -1, -1, 1, 0, "#Mne1:Tag1 #Mne2:Tag2")

19.12.4 OpenPrevious

"OpenPrevious" is an integrated scripting function that re-opens the WinStudio screen that was closed last.

Function	Group	Execution	Windows	Embedded	Thin Client
OpenPrevious	Screens	asynchronous	supported	supported	supported

Syntax

OpenPrevious({ | optNumX1, optNumY1, optNumX2, optNumY2 })

Inputs

optNumX1, optNumY1, optNumX2, optNumY2

The coordinates in pixels for the upper-left (X1,Y1) and lower-right (X2,Y2) corners of the screen.

These are optional parameters. If no values are specified, the default screen size and location are used. For further information, refer to [Screen attributes, page 156](#).

Returned value

This function returns one of the following values:

Value	Description
0	Successful
1	Error



This function cannot be used in combination with [tasks, page 57](#) or in the [global procedures, page 716](#) script.

Examples

Opening the previous screen using its default size and location:

Appendix: WinStudio-Internal Scripting Language

OpenPrevious()

Opening the previous screen in the top-left corner of the display and sized to 800x600:

OpenPrevious(0, 0, 800, 600)

19.12.5 ShowInplaceInput

This function shows a simple text input dialog at a specified location in the project client/viewer.

Function	Group	Execution	Windows	Embedded	Thin Client
ShowInplaceInput	Screens	asynchronous	supported	supported	supported

Syntax

```
ShowInplaceInput("tagOutput",numStartXPos,numStartYPos{ | ,optNumMin,optNumMax{ | ,
{optNumEnablePasswordMode | 0 | 1 }{ | ,{ optNumShowOSVK | 0 | 1 } } } }
```

Inputs**tagOutput**

The name of a tag that receives the input.



The tag name has to be enclosed in quotation marks as shown in the syntax diagram. Otherwise, the runtime tries to call the value of the specified tag.

numStartXPos

The starting X position in the top-left corner of the input dialog – i.e., the number of pixels between this corner and the left side of the display.

numStartYPos

The starting Y position of the top-left corner of the input dialog – i.e., the number of pixels between this corner and the top of the display.

optNumMin

The minimum numeric value that is accepted by the input dialog.

This parameter is optional. If no value is specified, the dialog accepts any value.

optNumMax

The maximum numeric value that is accepted by the input dialog.

This parameter is optional. If no value is specified, the dialog accepts any value.

optNumEnablePasswordMode

Option to enable Password mode, which hides the operator input as if it is a password:

Appendix: WinStudio-Internal Scripting Language

Value	Description
0	Show input as plain text
1	Hide input

This parameter is optional. If no value is specified, the default value is 0.

optNumShowOSVK

Option to show the default virtual keyboard [configured in the project settings, page 69](#):

Value	Description
0	Do not show the virtual keyboard
1	Show virtual keyboard

This parameter is optional. If no value is specified, the default value is 0.

Returned value

This function returns the following possible values:

Value	Description
0	Successful
-1	Invalid tag specified for tagOutput
-2	Invalid number of parameters
-3	Viewer is not running

Examples

ShowInplaceInput - Examples

```
ShowInplaceInput( "OperatorInput", 50, 50 )
```

```
ShowInplaceInput( "OperatorInput", 50, 50, 1, 100 )
```

```
ShowInplaceInput( "OperatorInput", 50, 50, 1, 100, 0, 1 )
```

19.12.6 ShowMessageBox

This function shows a simple message box with either an **OK** button or **Yes / No** buttons.

Function	Group	Execution	Windows	Embedded	Thin Client
ShowMessage-Box	Screens	synchronous	supported	supported	supported

Syntax

```
ShowMessageBox(strMessage{ | , { optNumButtons | 0 | 4 } { | , optStrTitle } })
ShowMessageBox( strMessage )
ShowMessageBox( strMessage, optNumButtons )
ShowMessageBox( strMessage, optNumButtons, optStrTitle )
```

Inputs

strMessage

The message body to be displayed in the box.

optNumButtons

A numeric flag that specifies which kind of confirmation buttons to display in the message box:

Appendix: WinStudio-Internal Scripting Language

Value	Description
0	OK button
4	Yes / No buttons



To add an exclamation mark to the window – to distinguish warnings and alarms from plain messages – add 48 (vbExclamation) to this parameter. For further information, refer to "Examples" below.

This parameter is optional. If no value is specified, the default value is 0.

optStrTitle

The title of the message.

This parameter is optional. If no value is specified, no title is displayed.

Returned value This function returns the following possible values:

Value	Description
1	Operator clicked on OK
6	Operator clicked on Yes
7	Operator clicked on No

Examples *ShowMessageBox - Examples*

```
ShowMessageBox( "The action could not be completed." )
```

```
ShowMessageBox( "Continue with action?", 4 )
```

```
ShowMessageBox( "The action could not be completed.", 0+48, "Alert" )
```

19.13 Security Functions

19.13.1 Security Functions – General Information

These functions are used to manage users and groups in the security system of projects.

Security

Function	Execution	Windows PC	Windows CE	Thin Client
BlockUser, page 535	synchronous	supported	supported	supported
CheckESign, page 535	synchronous	supported	supported	supported
CreateUser, page 536	synchronous	supported	supported	supported
ExportSecuritySystem, page 537	synchronous	supported	supported	supported
GetSecuritySystemStatus, page 538	synchronous	supported	supported	supported
GetUserFullName, page 539	synchronous	supported	supported	supported
GetUserNames, page 540	synchronous	supported	supported	executed on the server
GetUserPWDAging, page 541	synchronous	supported	supported	supported
GetUserState, page 541	synchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Function	Execution	Windows PC	Windows CE	Thin Client
RemoveUser, page 543	synchronous	supported	supported	supported
ImportSecuritySystem, page 542	synchronous	supported	supported	supported
SetPassword, page 544	synchronous	supported	supported	supported
UnblockUser, page 545	synchronous	supported	supported	supported

19.13.2 BlockUser

"BlockUser" is an internal scripting function that blocks an existing user from logging onto a project.

Function	Group	Execution	Windows	Embedded	Thin Client
BlockUser	Security	synchronous	supported	supported	supported

Syntax

```
BlockUser( strUserName )
```

Inputs

strUserName

The name of the user to block.

Returned value

This function returns the following possible values:

Value	Description
0	User blocked successfully
1	Invalid number of parameters
2	Incorrect parameter type
3	Specified user does not exist
4	User currently logged on does not have the rights to block (i.e., user does not have Edit Security System)
5	The operation on the distributed security system failed
6	User cannot be blocked
7	The current Security mode does not allow users to be blocked/unblocked
8	Internal error

Examples

Blocking the user named Bob:

```
BlockUser( "Bob" )
```

Blocking the user specified in position 3 of the "badUsers" array:

```
BlockUser( badUsers[3] )
```

Block the user that is currently logged on:

```
BlockUser( UserName )
```

19.13.3 CheckESign

"CheckESign" is an internal scripting function that prompts the runtime user to electronically sign an event by entering their user name and password. It can be called to secure specific expressions and scripts, just as the **E-Sign** option secures screen objects.

Appendix: WinStudio-Internal Scripting Language

Function	Group	Execution	Windows	Embedded	Thin Client
CheckESign	Security	synchronous	supported	supported	supported

Syntax

CheckESign()

This function does not have any parameters. On calling the function, a security dialog is displayed requesting the user to enter user name and password.

Returned value This function returns the following possible values:

Value	Description
0	User name and/or password not accepted
1	User name and password accepted



User names and passwords are stored in the [Security folder](#), [page 357](#).

Examples *CheckESign - Examples*

CheckESign()

19.13.4 CreateUser

This function creates a new user in the project security system.

Function	Group	Execution	Windows	Embedded	Thin Client
CreateUser	Security	synchronous	supported	supported	supported

Syntax

CreateUser(strUserName, strGroup, strPassw{ | , optStrUserFullName })

Inputs**strUserName**

The name of the user to be created

strGroup

The name of the group the user is to be assigned to

strPassw

The user password

optStrUserFullName

The full name of the user

Returned value This function returns the following possible values:

Value	Description
-1	Internal error; contact technical support
0	New user created successfully
1	Invalid number of parameters

Appendix: WinStudio-Internal Scripting Language

Value	Description
2	Incorrect parameter type
3	User name already exists
4	Group does not exist
5	Failed to save to configuration file
6	Invalid user
7	Full user name already exists



Users created with this function are not shown in the project "Security" folder because they are stored in a secondary database. To manage this database, use the "ExtUser.exe" program (located in the "Bin" subfolder of the application folder).

Examples *CreateUser - Examples*

```
CreateUser( "Bob", "Admin", "Chocolate", "Bob Smith" )
CreateUser( "Albert", "Engineering", "EMC2", "Albert Jones" )
```

19.13.5 ExportSecuritySystem

This function exports the security system configuration to an encoded file.

Function	Group	Execution	Windows	Embedded	Thin Client
ExportSecurity-System	Security	synchronous	supported	supported	supported

Syntax

```
ExportSecuritySystem(strFileName, strPassword)
```

Inputs

strFileName

The complete file path and name to save the configuration file.

strPassword

The main password for the security system. This same password is used to protect the exported file.

Returned value This function returns the following possible values:

Value	Description
-2	Incorrect parameter type
-1	Invalid number of parameters
0	Could not write security data
1	File exported successfully

Examples *ExportSecuritySystem - Examples*

```
ExportSecuritySystem( "C:\security.txt" )
ExportSecuritySystem( "C:\security.txt", "mypa55w0rd" )
```

Appendix: WinStudio-Internal Scripting Language

19.13.6 GetSecuritySystemStatus

This function gets the status of the security system and its connection to the authentication server if the security mode is either Distributed-Client or Domain (LDAP).

Function	Group	Execution	Windows	Embedded	Thin Client
GetSecuritySystemStatus	Security	synchronous	supported	supported	supported

Syntax

```
GetSecuritySystemStatus({ | { optNumType | 0 | 1 | 2 } })
```

```
GetSecuritySystemStatus()
```

```
GetSecuritySystemStatus( optNumType )
```

Inputs**optNumType**

Specifies the actions to be carried out in case of a status update.

Value	Description
0	Execute a fast check using either Ping or Bind (depending on the server settings), however, take no other actions
1	Force reload of users and groups from the authentication server
2	Clear cached users and groups

This is an optional parameter. If no value is specified, the default value is "0".

Returned value

This function returns the following possible values:

Value	Security Mode is Distributed-Client	Security Mode is Domain (LDAP)
0	No cache	Connection timeout
1	Updated cache	Bind timeout
2	Outdated local cache	Query timeout
3	Outdated server cache	Disconnected
4	Disconnected from server	Connected
5	N/A	Query returned no users or groups
6	N/A	Invalid user or group

Appendix: WinStudio-Internal Scripting Language



This function returns the same value that is sent to the project tag configured in the **Status Tag** field, in the [security system server settings, page 362](#).

Additionally, there are other results besides calling this function that update the status:

- If a user logs on to the project. Specifically, if the user logs on via the built-in "LogOn" dialog (opened by either calling the "LogOn" function or by selecting the **LogOn** menu command in the Viewer), the status is updated before the dialog is displayed.
- If the authentication server is offline and the retry interval (configured in the security system server settings) has elapsed.
- If the security system settings are opened in the development environment.

Whenever the status is updated, the new value is immediately sent to the project tag configured in the **Status Tag** field.

Examples

Getting the status of the security system:

GetSecuritySystemStatus()

Forcing the security system to reload all users and groups from the authentication server:

GetSecuritySystemStatus(1)

19.13.7 GetUserFullName

This function gets the full name (if specified) of a specified user in the project security system.

Function	Group	Execution	Windows	Embedded	Thin Client
GetUserFullName	Security	synchronous	supported	supported	supported

Syntax

```
GetUserFullName(strUserName, "tagUserFullName")
```

Inputs

strUserName

Name of a user in the project security system.

tagUserFullName

The name of a tag (String type) that is to receive the full name of the specified user after the function is called. If no full name is defined for the specified user, the tag receives an empty string ("").



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the function tries to get the value of the specified tag.

Returned value

This function returns the following possible values:

Appendix: WinStudio-Internal Scripting Language

Value	Description
0	Specified user does not exist
1	Successful; the specified user exists

Examples Getting the full name of the currently logged-on user (via the system tag "UserName"):

GetUserFullName(UserName, "UserFullName")

Getting the full name of the user "engineer1":

GetUserFullName("engineer1", "UserFullName")

19.13.8 GetUserNames

Function	Group	Execution	Windows	Embedded	Thin Client
GetUserNames	Security	synchronous	supported	supported	executed on the server

Syntax

```
GetUserNames( "tagUsers", optNumUserType, "opttagGroups" )
```

Inputs

tagUsers

Name of the array tag that is to receive the list of user names

optNumUserType

Type of the user to get

0	Returns all users
1	Only users created during runtime
2	Only users created using the development environment

opttagGroups

Name of the array tag that is to receive the list of group names for each user

Returned value The number of users or a negative number from the following list:

-1	Invalid number of parameters
-2	tagUsers is invalid
-3	optNumUserType is invalid
-4	opttagGroups is invalid
-5	Error, function cannot be called in the Thin Client

Examples

Tag name	Expression
NumberOfUsers	GetUserNames("UsersArray") // Retrieves all users, stores names in the UsersArray tag and the number of users in the NumberOfUsers tag.
NumberOfUsers	GetUserNames("UsersArray", 1) // Retrieves all users created during runtime, stores names in the UsersArray tag and the number of users in the NumberOfUsers tag.

Appendix: WinStudio-Internal Scripting Language

Tag name	Expression
NumberOfUsers	GetUserNames("UsersArray", 2) // Retrieves all users created in the development environment, stores names in the UsersArray tag and the number of users in the NumberOfUsers tag.
NumberOfUsers	GetUserNames("UsersArray", 2, "Groups") // Retrieves all users created in the development environment, stores names in the UsersArray tag and the number of users in the NumberOfUsers tag. The group name per each user is stored in the Groups tag.

19.13.9 GetUserPwdAging

Returns the time remaining before the password for a specific user expires.

Function	Group	Execution	Windows	Embedded	Thin Client
GetUserPwdAging	Security	synchronous	supported	supported	supported

Syntax

```
GetUserNames( strUser )
```

Inputs

strUser

User name of which the password aging is checked.

Returned value

≤0	Number of hours since the password expired
0	Specified user is not logged on
>0	Number of hours remaining until the password expires



If the function is not executed properly (e.g., User Name is invalid) or if the specified user is not logged on, the function returns BAD quality.

Examples

Tag name	Expression
TagHoursToExpire	GetUserPwdAging("John") // Returns the number of hours before the password for the User "John" expires.
TagHoursToExpire	GetUserPwdAging(UserName) // Returns the number of hours before the password for the current User logged on the system expires.

19.13.10 GetUserState

Gets the current status of a selected user.

Function	Group	Execution	Windows	Embedded	Thin Client
GetUserState	Security	synchronous	supported	supported	supported

Syntax

```
GetUserState(strUserName)
```

Appendix: WinStudio-Internal Scripting Language

Inputs

strUserName

The name of the user.

Returned value

This function returns the following possible values:

Value	Description
-3	Specified user does not exist
-2	Incorrect parameter type
-1	Invalid number of parameters
0	Specified user is unblocked
1	Specified user is blocked

Examples

Tag name	Expression
Tag	GetUserState("Bob")
Tag	GetUserState ("Albert")

19.13.11 ImportSecuritySystem

This function imports a security system configuration from an external file.

Function	Group	Execution	Windows	Embedded	Thin Client
ImportSecurity-System	Security	synchronous	supported	supported	supported

Syntax

```
ImportSecuritySystem(strSecuritySystemPassword, strFileName, strFilePassword{ | , optNumMode } )
```

Inputs

strSecuritySystemPassword

The main password for the current security system configuration of the project. (The security system has to be enabled.)

strFileNameThe complete file path and name of the configuration file to be imported. (Before, the file has to be exported from a "WinStudio" project using either the [Security System configuration tool, page 377](#) or the [ExportSecuritySystem, page 537](#) function.)**strFilePassword**

The password for the specified configuration file.

optNumMode

A numeric value specifying how the imported settings are to be handled:

Appendix: WinStudio-Internal Scripting Language

Value	Description
0	Attaching the imported settings to the current settings. In the event of a conflict, the current settings are replaced by the imported settings.
1	Attaching the imported settings to the current settings. In the event of a conflict, the current settings are kept.
2	Replace the current settings completely by the imported settings.

This is an optional parameter. If no value is specified, the default value is "0".

Returned value This function returns the following possible values:

Value	Description
-2	Incorrect parameter type
-1	Invalid number of parameters
0	Could not read security data
1	File imported successfully

Examples *Program:*

```
ExportSecuritySystem( "curr3ntPa55w0rd", "C:\security.txt", "1mp0rtPa55w0rd" )
```

```
ExportSecuritySystem( "curr3ntPa55w0rd", "C:\security.txt", "1mp0rtPa55w0rd", 2 )
```

19.13.12 RemoveUser

Removes a user from the WinStudio security system.

Function	Group	Execution	Windows	Embedded	Thin Client
RemoveUser	Security	synchronous	supported	supported	supported

Syntax

```
RemoveUser( strUserName )
```

Inputs

strUserName

The name of the user to be removed.

Returned value This function returns the following values:

Value	Description
0	User removed successfully
1	Invalid number of parameters
2	Incorrect parameter type
3	User currently logged on does not have the rights to remove (i.e., user does not have Edit Security System)
4	User cannot be removed
5	Specified user does not exist
6	Component-level failure
7	Failed to save to configuration file
8	The current Security mode does not allow user to be removed

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	RemoveUser("Bob")
Tag	RemoveUser("Albert")



This function can only be used to remove users created using the [CreateUser\(\), page 536](#) function.

19.13.13 SetPassword

Specifies a new password for a user.

Function	Group	Execution	Windows	Embedded	Thin Client
SetPassword	Security	synchronous	supported	supported	supported

Syntax

```
SetPassword( strUserName, optStrNewPassword )
```

Inputs

strUserName

The name of the user.

optStrNewPassword

"Optional" The new password



If this parameter is omitted, the "SetPassword" function displays an "Enter a new password" dialog to enter a new password.

Returned value

This function returns a numeric value with the following meaning:

Value	Description
-1	Internal error; contact technical support
0	Password set successfully
1	Invalid number of parameters
2	Incorrect parameter type
3	Specified user does not exist
4	Re-entrant call not allowed
5	User clicked "Cancel"
6	Group does not exist
7	Password too weak
8	Invalid password.
9	Invalid user

Examples

Tag name	Expression
Tag	SetPassword("Bob")
Tag	SetPassword("Albert", "anemarie")

19.13.14 UnblockUser

This function unblocks a blocked user in the security system.

Function	Group	Execution	Windows	Embedded	Thin Client
UnblockUser	Security	synchronous	supported	supported	supported

Syntax

```
UnblockUser( strUserName )
```

Inputs

strUserName

The name of the user to be unblocked.

Returned value

This function returns the following possible values:

Value	Description
0	User successfully unblocked
1	Invalid number of parameters
2	Incorrect parameter type
3	Specified user does not exist
4	User currently logged on does not have the rights to unblock (i.e., user does not have Edit Security System)
5	The operation on the distributed security system failed
6	Specified user cannot be blocked
7	The current Security mode does not allow users to be blocked/unblocked
8	Internal error

Examples

Tag name	Expression
Tag	UnblockUser("Bob")
Tag	UnblockUser("Albert")

19.14 Module Activity Functions

19.14.1 Module Activity Functions – General Information

These functions are used to manage the various runtime modules of a project, e.g., background tasks, the data server, and the project viewer, as well as module interaction with the operating system.

Module activity

Function	Execution	Windows PC	Windows CE	Thin Client
AppActivate, page 547	asynchronous	supported	supported	supported
AppIsRunning, page 548	synchronous	supported	supported	supported
AppPostMessage, page 549	synchronous	supported	supported	supported
AppSendKeys, page 550	synchronous	supported	not supported	supported

Appendix: WinStudio-Internal Scripting Language

Function	Execution	Windows PC	Windows CE	Thin Client
CleanReadQueue, page 550	synchronous	supported	not supported	executed on the server
CloseSplashWindow, page 550	synchronous	supported	not supported	executed on the server
DisableMath, page 551	asynchronous	supported	supported	executed on the server
EnableMath, page 551	asynchronous	supported	supported	executed on the server
EndTask, page 551	asynchronous	supported	supported	executed on the server
ExitWindows, page 552	asynchronous	supported	not supported	supported
IsScreenOpen, page 553	synchronous	supported	supported	not supported
IsTaskRunning, page 553	synchronous	supported	supported	executed on the server
IsViewerInFocus, page 554	synchronous	supported	not supported	not supported
KeyPad, page 554	asynchronous	supported	supported	supported
LogOff, page 556	asynchronous	supported	supported	supported
LogOn, page 556	asynchronous	supported	supported	supported
Math, page 557	synchronous	supported	supported	executed on the server
PostKey, page 557	synchronous	supported	supported	supported
Recipe, page 558	synchronous	supported	supported	executed on the server
Report, page 558	synchronous	supported	supported	executed on the server
RunGlobalProcedureOnServer, page 564	synchronous	supported	supported	supported
RunVBScript, page 566	synchronous	supported	supported	supported
SecureViewerReload, page 567	synchronous	not supported	not supported	supported (Secure Viewer only)
SendKeyObject, page 568	synchronous	supported	not supported	not supported
SetAppPath, page 570	synchronous	supported	not supported	executed on the server
SetViewerInFocus, page 570	synchronous	supported	not supported	supported
SetViewerPos, page 571	synchronous	supported	not supported	supported
ShutDown, page 572	synchronous	supported	supported	not supported
StartTask, page 572	asynchronous	supported	supported	executed on the server
ViewerPostMessage, page 573	asynchronous	supported	supported	supported
Wait, page 574	synchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Function	Execution	Windows PC	Windows CE	Thin Client
WinExec, page 574	asynchronous	supported	supported	supported
WinExecIsRunning, page 576	synchronous	supported	supported	supported

19.14.2 AppActivate

"AppActivate" is an internal scripting function that activates (i.e., brings to the front) another application window that is already open.

Function	Group	Execution	Windows	Embedded	Thin Client
AppActivate	Module activity	asynchronous	supported	supported	supported

Syntax

```
AppActivate(strAppTitle{ | , { optNumActiv | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 } { | , optNumTimeout } } )
```

Inputs

strAppTitle

The full title (as shown in the title bar) of the application window.

optNumActiv

Controls how the specified window is to be activated:

Value	Command	Description
0	SW_HIDE	Hides the currently active window and then activates the specified window.
1	SW_SHOWNORMAL	Activates and displays the specified window. If the window is minimized or maximized, it is restored to its original size and position. This command is to be used if a window is displayed for the first time.
2	SW_SHOWMINIMIZED	Activates the specified window and then minimizes it.
3	SW_SHOWMAXIMIZED	Activates the specified window and then maximizes it.
4	SW_SHOWNOACTIVATE	Displays the specified window but does not activate it. If the window is minimized or maximized, it is restored to its original size and position.
5	SW_SHOW	Activates and displays the specified window in its current size and position. This function is similar to SW_SHOWNORMAL except that if the window is minimized or maximized, it remains in that state.
6	SW_MINIMIZE	Minimizes the specified window and then activates the next open window.
7	SW_SHOWMINNOACTIVATE	Displays the specified window as a minimized window but does not activate it.
8	SW_SHOWNA	Displays the specified window in its current size and position but does not activate it. This function is similar to SW_SHOWNOACTIVATE except that if the window is minimized or maximized, it remains in that state.
9	SW_RESTORE	Activates and displays the specified window. If the window is minimized or maximized, it is restored to its original size and position. This command is to be used when restoring a minimized window.

This parameter is optional. If no value is specified, the default command is "SW_RESTORE".

Appendix: WinStudio-Internal Scripting Language

optNumTimeout

The timeout period (in milliseconds) for the function to be successfully executed. If the function is not executed in this period for any reason, it is aborted.

This parameter is optional. If no value is specified, the default timeout is five seconds (or 5000 milliseconds).

Returned value This function returns one of the following values:

Value	Description
0	ERROR: The specified application window was not activated or did not respond within the specified timeout period.
1	SUCCESS: The specified application window was successfully activated.



"AppActivate" is similar to the function [ShowWindow](#) in the Microsoft Windows API and enables many of the same options. For further information, please refer to the Windows API documentation.

Examples Displaying the Microsoft Word document named "test.doc":

AppActivate("test.doc — Microsoft Word", 5)

19.14.3 AppIsRunning

"AppIsRunning" is an internal scripting function that checks if another application window (e.g., a Microsoft Word document) is open and running.

Function	Group	Execution	Windows	Embedded	Thin Client
AppIsRunning	Module activity	synchronous	supported	supported	supported

Syntax

```
AppIsRunning(strAppTitle{ | ,optNumTimeout })
```

Inputs**strAppTitle**

The full title (as shown in the title bar) of the application window.

optNumTimeout

The timeout period (in milliseconds) for the function to be successfully executed. If the function is not executed in this period for any reason, it is aborted.

This parameter is optional. If no value is specified, the default timeout is five seconds (or 5000 milliseconds).

Returned value This function returns one of the following values:

Value	Description
0	ERROR: The specified application window is not open or did not respond within the specified timeout period.
1	SUCCESS: The specified application window is open and running.



"AppIsRunning" is similar to the function [IsWindow](#) in the Microsoft Windows API. For further information, please refer to the Windows API documentation.

Appendix: WinStudio-Internal Scripting Language

Examples Verifying that the Microsoft Word document named "test.doc" is open and running:

ApplsRunning("test.doc — Microsoft Word")

19.14.4 AppPostMessage

"AppPostMessage" is an internal scripting function that sends a Windows system message to another application window.

Function	Group	Execution	Windows	Embedded	Thin Client
AppPostMessage	Module activity	asynchronous	supported	supported	supported

Syntax

```
AppPostMessage(strAppTitle, strMessage, numWParam, numLParam{ | , optNumTimeout } )
```

Inputs

strAppTitle

The full title (as shown in the title bar) of the application window.

strMessage

The name or code of the system message.



The "CLOSE", "MINIMIZE", "MAXIMIZE" and "RESTORE" messages can be entered as string values enclosed in quotation marks. All other message codes have to be given as numeric values.

numWParam

Additional message-specific information.

numLParam

Additional message-specific information.

optNumTimeout

The timeout period (in milliseconds) for the function to be successfully executed. If the function is not executed in this period for any reason, it is aborted.

This parameter is optional. If no value is specified, the default timeout is five seconds (or 5000 milliseconds).

Returned value

This function returns one of the following values:

Value	Description
0	ERROR: The specified application window is not open or did not respond within the specified timeout period.
1	SUCCESS: The specified application window is open and running.



"AppPostMessage" is similar to the function [PostMessage](#) in the Microsoft Windows API and enables many of the same options. For further information including a list of available system messages, please refer to the Windows API documentation.

Appendix: WinStudio-Internal Scripting Language

Examples Closing the Calculator application:

AppPostMessage("Calculator", "CLOSE", 3, 1)

19.14.5 AppSendKeys

Sends keyboard commands to the active application.

Function	Group	Execution	Windows	Embedded	Thin Client
AppSendKeys	Module activity	synchronous	supported	not supported	supported

Syntax

```
AppSendKeys( strKeys1, strKeys2, ... , strKeysN )
```

Inputs

strKeys (1-N)

String tags containing the keyboard commands to be used.

Returned value No returned values.

Examples

Tag name	Expression
	AppSendKeys("S", "t", "u", "d", "i", "o", "<ENTER>")
	AppSendKeys("<Alt>F")



<ALT>, <CTRL>, or <SHIFT> can be specified in the text to send a code equal to the "Alt", "Ctrl" or "Shift" keyboard commands. To send the "<" character, specify "<<" in the text.

19.14.6 CleanReadQueue

Removes all reading messages from the communications module.

Function	Group	Execution	Windows	Embedded	Thin Client
CleanReadQueue	Module activity	synchronous	supported	not supported	executed on the server

Syntax

```
CleanReadQueue( )
```

This function does not have any parameters.

Returned value No returned values.

Examples

Tag name	Expression
	CleanReadQueue()



This function is not to be used in new projects. For compatibility reasons, however, it is still available for "WinStudio" projects created with previous versions.

19.14.7 CloseSplashWindow

Closes the "WinStudio" splash screen.

Function	Group	Execution	Windows	Embedded	Thin Client
CloseSplash-Window	Module activity	synchronous	supported	not supported	executed on the server

Appendix: WinStudio-Internal Scripting Language

Syntax

```
CloseSplashWindow()
```

This function does not have any parameters.

Returned value

No returned values.

Examples

Tag name	Expression
	CloseSplashWindow()

19.14.8 DisableMath

"DisableMath" is an internal scripting function that stops the execution of all math worksheets.

Function	Group	Execution	Windows	Embedded	Thin Client
DisableMath	Module activity	asynchronous	supported	supported	executed on the server

Syntax

```
DisableMath()
```

This function does not have any parameters.

Returned value

There are no returned values for this function.



To resume the execution of [math worksheets, page 324](#), call the [EnableMath, page 551](#) function.

Examples

Tag name	Expression
	DisableMath()

19.14.9 EnableMath

"EnableMath" is an internal scripting function that reactivates the execution of all math worksheets.

Function	Group	Execution	Windows	Embedded	Thin Client
EnableMath	Module activity	asynchronous	supported	supported	executed on the server

Syntax

```
EnableMath()
```

This function does not have any parameters.

Returned value

There are no returned values for this function.



In most cases, execution was paused by calling the [DisableMath, page 551](#) function.

Examples

Tag name	Expression
	EnableMath()

19.14.10 EndTask

Terminates the "WinStudio" module that is currently running.

Appendix: WinStudio-Internal Scripting Language

Function	Group	Execution	Windows	Embedded	Thin Client
EndTask	Module activity	asynchronous	supported	supported	executed on the server

Syntax

EndTask(strTask)

Inputs**strTask**

The name of the task to be stopped (from the following list):

BGTASK	Background tasks
VIEWER	Viewer
DBSPY	Database Spy
LOGWIN	LogWin
DRIVERDriver-Name	Driver
UNIDDECL	DDE client
UNINDDE	DDE server
UNIODBC	ODBC
TCPSERVER	TCP/IP server
TCPCLIENT	TCP/IP client
OPCCLIENT	OPC

Returned value

No returned values.

Examples

Tag name	Expression
	EndTask("Viewer")



To close a driver, the following syntax is to be used:

EndTask("DriverDriverName")

Where DriverName is the name of the .dll file of the driver. For example:

EndTask("DriverMODBU")

19.14.11 ExitWindows

Terminates the Windows operating system in a specified manner.

Function	Group	Execution	Windows	Embedded	Thin Client
ExitWindows	Module activity	asynchronous	supported	not supported	supported

Syntax

ExitWindows(numExitCode)

Inputs**numExitCode**

A numeric code specifying how Windows is to be terminated:

Appendix: WinStudio-Internal Scripting Language

Value	Description
0	Restart
1	Log off
2	Shut down

Returned value No returned values.

Examples

Tag name	Expression
	ExitWindows(1)

19.14.12 IsScreenOpen

Checks if a WinStudio screen is open.

Function	Group	Execution	Windows	Embedded	Thin Client
IsScreenOpen	Module activity	asynchronous	supported	supported	not supported



This function cannot be used in combination with [tasks, page 57](#) or in the [global procedures, page 716](#) script.

Syntax

```
IsScreenOpen(strScreen{ | ,optNumID })
```

Inputs

strScreen

The name of the WinStudio screen to be checked.

optNumID

The unique instance number of the screen. (The number is assigned when the screen is opened with the [Open, page 529](#) function.)

This parameter is optional. If no value is specified, the default ID is 0.

Returned value

0	The screen is not open.
1	The screen is open.

Examples

Tag name	Expression
Tag	IsScreenOpen("main") // Is "main" screen open?
Tag	IsScreenOpen("main", 10) // Is "main" screen with ID 10 open?

19.14.13 IsTaskRunning

Checks whether a background task is running.

Function	Group	Execution	Windows	Embedded	Thin Client
IsTaskRunning	Module activity	asynchronous	supported	supported	executed on the server

Syntax

```
IsTaskRunning(strTask)
```

Appendix: WinStudio-Internal Scripting Language

Inputs

strTask

The name of the task to be checked.

Returned value

0	Task is not running.
1	Task is running.

Examples

Is the project viewer (**Viewer.exe**) running?**IsTaskRunning("Viewer")**

19.14.14 IsViewerInFocus

Checks whether the project viewer ("Viewer.exe") is in focus on the screen.

Function	Group	Execution	Windows	Embedded	Thin Client
IsViewerInFocus	Module activity	synchronous	supported	not supported	not supported

Syntax

IsViewerInFocus()

This function does not have any parameters.

Returned value

0	Viewer is not in focus.
1	Viewer is in focus.

Examples

IsViewerInFocus - Examples

IsViewerInFocus()

19.14.15 KeyPad

Displays a virtual keyboard to prompt the runtime user to enter a tag value.

Function	Group	Execution	Windows	Embedded	Thin Client
KeyPad	Module activity	asynchronous	supported	supported	executed on the server

Syntax

KeyPad("strTagName", optStrKeyboardName, optNumIsPassword, optStrHint, optNumMin, optNumMax, optNumESign, optStrConfirmation)

Inputs

strTagName

The name of the tag that is to be written to by means of the "Virtual Keyboard".

This name has to be enclosed in quotation marks. Otherwise, the project tries to get the contents of the tag.

optStrKeyboardName

Type of "Virtual Keyboard" (e.g., "AlphaNumeric", "Enhanced Keypad" or "Keypad"). If this parameter is omitted, the default "Virtual Keyboard" is launched.

Appendix: WinStudio-Internal Scripting Language

optNumIsPassword

If this parameter is set at any value different from 0 (zero), the characters typed in the "Virtual Keyboard" appear as asterisks ("*****"). This option is useful if the user is entering a password.

optStrHint

The value specified for this parameter is displayed in the title bar of the "Virtual Keyboard" window if the **Show Hint** option is selected in the [project settings, page 69](#).

optNumMin, optNumMax

Numeric minimum and maximum values for the tag when using the keyboard type "Keypad" if the option **Show MIN/MAX fields** is enabled in the [project settings, page 69](#). (These values are ignored for all other keyboard types.) These are optional parameters. For implementation, however, both have to be specified. If only one parameter is specified – e.g., Min but not Max – it is ignored.

optNumESign

If this parameter is set with any value different from 0 (zero), the user is prompted to log on to complete the input and the action is noted in the log.

optStrConfirmation

The value specified for this parameter is displayed in a confirmation dialog that has to be confirmed by the user to complete the input.



Confirmation cannot be automated or bypassed. The dialog can only be confirmed by the user actually pressing a key or mouse button.

This parameter is optional. If no value is specified, no dialog is displayed.

Returned value

Error	Description
0	Successful
1	Error
2	Tag does not exist
3	Reentrant error, function is already running
4	Invalid number of parameters
5	Internal error, contact the technical support for more information

Examples

Tag name	Expression
Tag	KeyPad("tagA")
Tag	KeyPad("tagA", "EnhKeypad")
Tag	KeyPad("tagA", "EnhKeypad", 1)
Tag	KeyPad("tagA", "EnhKeypad", 1, "My Input", 0, 100)

Appendix: WinStudio-Internal Scripting Language

19.14.16 LogOff

This function logs off the current user and logs on the default user (typically "guest").

Function	Group	Execution	Windows	Embedded	Thin Client
LogOff	Module activity	asynchronous	supported	supported	supported

Syntax

```
LogOff()
```

This function does not have any parameters.

Returned value This function returns the following possible values:

Value	Description
0	Error
1	Successful

Examples *LogOff - Examples*

```
LogOff()
```

19.14.17 LogOn

This function either logs on a specified user or displays a "Log On" dialog.

Function	Group	Execution	Windows	Embedded	Thin Client
LogOn	Module activity	asynchronous	supported	supported	supported

Syntax

```
LogOn({ | optStrUsername, optStrPassword })
```

Inputs**optStrUsername**

Name of the user to be logged on.

optStrPassword

Password of the specified user.

"optStrUsername" and "optStrPassword" are optional parameters. If they are not specified, the project displays a [Log On dialog, page 378](#) instead to prompt the current operator of the station to log on.

Returned value This function returns the following possible values:

Value	Description
0	Error (e.g., user name or password is invalid) or cancellation
1	Successful

Examples Displaying a Log On dialog:

```
LogOn()
```

Username "Albert" with password "EMC2":

```
LogOn( "Albert", "EMC2" )
```

19.14.18 Math

Executes the specified math worksheet.

Function	Group	Execution	Windows	Embedded	Thin Client
Math	Module activity	synchronous	supported	supported	executed on the server

Syntax

Math(numWorksheet)

Inputs

numWorksheet

Number of the math worksheet to be executed.

Returned value

No returned values.

Examples

Tag name	Expression
	Math(6)

NOTICE

Running a math worksheet from inside another module pauses that module until the math worksheet finishes.

For this reason, this function is only to be used if it is absolutely necessary to avoid decreasing the performance of the other modules.

19.14.19 PostKey

Sends a keyboard code to the currently displayed WinStudio screen.

Function	Group	Execution	Windows	Embedded	Thin Client
PostKey	Module activity	synchronous	supported	supported	supported



This function cannot be used in combination with [tasks, page 57](#) or in the [global procedures, page 716](#) script.

Syntax

PostKey(numKeyDownOrKeyup, numwParam, numlParam)

Inputs

numKeyDownOrKeyup

Numeric tag containing a 0 (to indicate a KeyDown event) or a 1 (to indicate a KeyUp event).

numwParam

Numeric tag containing the keyboard code to be sent.

numlParam

Numeric tag with the value of the "lParam" parameter of a Windows message.

Returned value

No returned values.

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
	PostKey(0, 0x24, 0)

19.14.20 Recipe

Executes the specified recipe worksheet.

Function	Group	Execution	Windows	Embedded	Thin Client
Recipe	Module activity	synchronous	supported	supported	supported

Syntax

```
Recipe( strFunction )
```

Inputs

strFunction

String tag specifying the action to be performed and the recipe worksheet to be used

in the **[action]:[recipe sheet]** format.

Actions:

Save	Saves data to a data file.
Load	Loads data from a data file
Delete	Deletes a data file
Init	Initializes a data file with a value of 0 in all tags

Returned value

0	No error
1	If the tag is numeric
2	Expression does not contain ":"
3	Previous command to the invalid ":"
4	Task not found by the system
5	Disk error

Examples

Tag name	Expression
Tag	Recipe("Save:Recipe1")
Tag	Recipe("Load:Recipe5")



- The Background task has to be running to execute the recipe functions.
- If this function is called on a Thin Client, the command is sent to the server (via TCP/IP) and the [Recipe, page 319 task, page 319](#) on the server executes the command. For this reason, please note that tags configured with a Scope of **Local** rather than **Server** are still updated on the server.

19.14.21 Report

Executes the specified report worksheet and sends the output to the hard disk, printer or a PDF file.

Appendix: WinStudio-Internal Scripting Language

Function	Group	Execution	Windows	Embedded	Thin Client
Report	Module activity	synchronous	supported	supported	supported

Syntax

```
Report( "strFunction" , optNumOrientation )
```

Inputs

strFunction

String specifying the output type to be performed and the report worksheet to be put out according to the syntax

"Output type:file name", where...

Output type is either **Disk** (saves data file to hard disk), **Prn** (sends report to default printer) or **Pdf** (generates a PDF file of the report) and **File name** is the name of the report worksheet file (*.rep) to put out.

optNumOrientation

Set the page orientation as follows: "0" (default) for Portrait, "1" for Landscape. This parameter is ignored if the **Operation** is other than **Prn**.



Some features of this function are not supported when running the project on a Windows CE device: it cannot generate PDFs; it cannot change paper orientation using the **optNumOrientation** parameter; and it does not support report worksheets in RTF format.

Returned value

Value	Description
0	Successful
1	strFunction is configured with a numeric value (invalid)
2	strFunction does not contain ""." (invalid)
3	strFunction contains an invalid output type before the ""."
4	Background task is not running (see hint below)
5	Disk error (e.g., disk full, read-only file cannot be overwritten or invalid path)
6	Specified report worksheet does not exist



The Background task has to be running in order to execute this function. Otherwise, the function returns the error code "4". For further information, refer to <Task execution>.

Examples

Tag name	Expression
	Report("Disk:Report1.rep")
	Report("Prn:Report2.rep", 0)
	Report("Prn:Report3.rep", 1)
	Report("Pdf:Report1.rep")

Appendix: WinStudio-Internal Scripting Language

19.14.22 RunGlobalProcedureAsync

This function executes a global procedure asynchronously in its own thread to ensure that other running scripts are not affected. The procedure is run on the project server, however, it can be called by any local or remote client.

Function	Group	Execution	Windows	Embedded	Thin Client
RunGlobalProcedureAsync	Module activity	asynchronous	supported	supported	not supported; see note

Syntax

```
RunGlobalProcedureAsync(strProcedureName{ | , optStrArgument1, ..., optStrArgumentN })
```

```
RunGlobalProcedureAsync( strProcedureName )
```

```
RunGlobalProcedureAsync( strProcedureName, optStrArgument1, ..., optStrArgumentN )
```

Inputs**strProcedureName**

The name of the procedure (i.e., a VBScript function or sub-routine defined in the [Procedures, page 716](#) folder) to run asynchronously.

optStrArgument1, ..., optStrArgumentN

Values that are transmitted to the parameters of the procedure. Arguments have to be transmitted as strings.

Returned value

If the procedure is successfully executed, this function returns a thread ID that can be used with the [RunGlobalProcedureAsyncGetStatus, page 561](#) function. Otherwise, this function returns an error code:

Value	Description
-1	Function is not supported on Viewer / Web Thin Client
-2	Invalid number of parameters At least the procedure name has to be specified
-3	Maximum number of threads exceeded See note
-4	Failed to compile VBScript parameters for execution
-5	Failed to start the thread execution
-100	Internal error Please contact technical support

Appendix: WinStudio-Internal Scripting Language



It has to be observed that this function can only be called by Background tasks (e.g., Math, Script, Scheduler) on the project server. It cannot be directly called from any project client, even if the project client is running on the same workstation or device as the project server, because the client process is single-threaded. To indirectly call the function from a project client, configure a [math, page 324](#) or [script, page 324](#), worksheet that can be executed via release (e.g., variable). By projecting an input object in a screen, the execution of the program code can be started manually.

Also, the maximum number of VBScript threads that can be executed asynchronously is configured by manually editing the project file (i.e., **projectname.APP**) and changing the following setting:

[Script]

MaxAsyncThreads=8

A value of 8 threads is set by default. This is only restricted by the system resources. Increasing the number of threads may decrease runtime performance.

Examples Given the following procedure that is defined in the Procedures folder...

RunGlobalProcedureAsync - Examples

```
Function AddMe(intNumber)
  If intNumber >= 6 Then
    AddMe = 0
  Else
    AddMe = intNumber + 2
  End If
End Function
```

...the procedure is run by calling the **RunGlobalProcedureAsync** function...

RunGlobalProcedureAsync("AddMe", "2")

...and the function returns a thread ID that can be used with the **RunGlobalProcedureAsyncGetStatus** function.

19.14.23 RunGlobalProcedureAsyncGetStatus

This function gets the status of one or more global procedures that were executed asynchronously by calling the "RunGlobalProcedureAsync" function. Each procedure is executed in its own thread so that other threads are not affected.

Function	Group	Execution	Windows	Embedded	Thin Client
RunGlobalProcedureAsyncGetStatus	Module activity	synchronous	supported	supported	not supported; see note

Syntax

```
RunGlobalProcedureAsyncGetStatus({ | optNumThreadID | "optTagThreadIDs", "optTagStatus", "optTagParameters"})
```

```
RunGlobalProcedureAsyncGetStatus( )
RunGlobalProcedureAsyncGetStatus( optNumThreadID )
RunGlobalProcedureAsyncGetStatus("optTagThreadIDs", "optTagStatus", "optTagParameters")
```

Inputs

optNumThreadID

The thread ID returned by the [RunGlobalProcedureAsync, page 560](#) function if the procedure was successfully executed.

Appendix: WinStudio-Internal Scripting Language

optTagThreadIDs

The name of an array tag that is to receive the thread IDs of all currently running and recently terminated threads.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

optTagStatus

The name of an array tag that is to receive the statuses of all currently running and recently terminated threads.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

optTagParameters

The name of an array tag that is to receive the parameters of all currently running and recently terminated threads.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

If the function succeeds, the possible returned values depend on how the function was called:

- If the function was called with no parameters...

RunGlobalProcedureAsyncGetStatus()

...the returned value is the total number of threads that are currently running.

- If the function was called with only the **optNumThreadID** parameter...

RunGlobalProcedureAsyncGetStatus(optNumThreadID)

...the returned value is either "0", indicating that the thread is still running or the value that was returned by the procedure.

- If the function was called with the array tags...

RunGlobalProcedureAsyncGetStatus(" optTagThreadIDs ", " optTag-Status ", " optTagParameters")

...the tags receive the appropriate values for all currently running and recently terminated threads.

If the function fails, it returns one of the following error codes:

Value	Description
-1	Function is not supported on Viewer / Web Thin Client
-2	Invalid thread ID
-3	Invalid optTagThreadIDs
-4	Invalid optTagStatus

Appendix: WinStudio-Internal Scripting Language

Value	Description
-5	Invalid optTagParameters
-100	Internal error Please contact technical support



It has to be observed that this function can only be called by Background tasks (e.g., Math, Script, Scheduler) on the project server. It cannot be directly called from any project client, even if the project client is running on the same workstation or device as the project server, because the client process is single-threaded. To indirectly call the function from a project client, configure a [math, page 324](#) or [script, page 324](#), worksheet that can be executed via release (e.g., variable). By projecting an input object in a screen, the execution of the program code can be started manually.

Also, if **RunGlobalProcedureAsync** is successfully called, an ID is returned for the created thread and the procedure is executed in that thread. The status of the thread is stored in an internal buffer and can be retrieved using the **RunGlobalProcedureAsyncGetStatus** function. The buffer is deleted if:

- The **RunGlobalProcedureAsyncGetStatus** function has been called and the thread status is different from "0" (thread is running); or
- The maximum buffer size has been exceeded, the thread is no longer running and a call to start a new thread has been made.

19.14.24 RunGlobalProcedureOnFalse

This function directly executes a global procedure if the value of a specified tag/expression becomes FALSE.

Function	Group	Execution	Windows	Embedded	Thin Client
RunGlobalProcedureOnFalse	Module activity	synchronous	supported	supported	supported

Syntax

```
RunGlobalProcedureOnFalse("strCondition", strProcedureOnFalse)
```

Inputs

strCondition

A project tag or expression



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

strProcedureOnFalse

The name of the procedure (i.e., a VBScript function or sub-routine defined in the [Procedures, page 716](#) folder) to run if the value of **strCondition** becomes FALSE (zero).

Returned value

This function returns the following possible values:

Appendix: WinStudio-Internal Scripting Language

Value	Description
0	Error
1	Successful



If this function is called, it remains active until the runtime environment is terminated. That means every time the tag/expression becomes FALSE, the procedure is executed. If the tag/expression becomes FALSE, the function is not executed continuously, however, only once "while" the tag/expression is FALSE.

The procedure is executed on the client. To execute a procedure on the server, use the [RunGlobalProcedureOnServer](#), page 564 function.

Examples If the value of the variable **TagOnFalse** becomes FALSE (zero), the procedure **UsingOnFalse** is executed:

RunGlobalProcedureOnFalse("TagOnFalse", "UsingOnFalse")

19.14.25 RunGlobalProcedureOnServer

This function directly executes a global procedure with parameters. The procedure is run on the project server, however, it can be called by any local or remote client.

Function	Group	Execution	Windows	Embedded	Thin Client
RunGlobalProcedureOnServer	Module activity	synchronous	supported	supported	supported

Syntax

```
RunGlobalProcedureOnServer(strProcedureName[ | ,optStrArgument1,...,optStrArgumentN ])
```

```
RunGlobalProcedureOnServer( strProcedureName, optStrArgument1, ..., optStrArgumentN )
```

Inputs

strProcedureName

The name of the procedure (i.e., a VBScript function or sub-routine defined in the [Procedures](#), page 716 folder) to be executed on the server.

optStrArgument1, ..., optStrArgumentN

Values that are transmitted to the parameters of the procedure. Arguments have to be transmitted as strings.

Returned value This function returns the value that is returned by the called procedure.

Examples Given the following procedure that is defined in the Procedures folder...

RunGlobalProcedureOnServer - Examples

```
Function AddMe(intNumber)
  If intNumber >= 6 Then
    AddMe = 0
  Else
    AddMe = intNumber + 2
  End If
End Function
```

...the procedure is run by calling the **RunGlobalProcedureOnServer** function...

RunGlobalProcedureOnServer("AddMe", "2")

Appendix: WinStudio-Internal Scripting Language

...and returns a value of 4.

19.14.26 RunGlobalProcedureOnTrigger

This function directly executes a global procedure if the value of a specified tag changes.

Function	Group	Execution	Windows	Embedded	Thin Client
RunGlobalProcedureOnTrigger	Module activity	synchronous	supported	supported	supported

Syntax

```
RunGlobalProcedureOnTrigger("strTagName", strProcedureOnTrigger)
```

Inputs

strTagName

Name of a project tag that calls a function if a value is changed.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

strProcedureOnTrigger

The name of the procedure (i.e., a VBScript function or sub-routine defined in the [Procedures, page 716](#) folder) to run if the value of the specified tag changes.

Returned value

This function returns the following possible values:

Value	Description
0	Error
1	Successful



If this function is called, it remains active until the runtime system is terminated. That means that every time the value of the tag changes, the procedure is executed.

The procedure is executed on the client. To execute a procedure on the server, use the [RunGlobalProcedureOnServer, page 564](#) function.

Examples

If the value of **TagTrigger** changes, the **UsingTrigger** procedure is to be executed:

```
RunGlobalProcedureOnTrigger("TagTrigger", "UsingTrigger")
```

19.14.27 RunGlobalProcedureOnTrue

This function directly executes a global procedure if the value of a specified tag/expression becomes TRUE.

Function	Group	Execution	Windows	Embedded	Thin Client
RunGlobalProcedureOnTrue	Module activity	synchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Syntax

RunGlobalProcedureOnTrue("strCondition",strProcedureOnTrue)

Inputs**strCondition**

A project tag or expression to trigger the function. The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

strProcedureOnTrue

The name of the procedure (i.e., a VBScript function or sub-routine defined in the [Procedures, page 716](#) folder) to run if the value of "strCondition" becomes TRUE (not zero).

Returned value

This function returns the following possible values:

Value	Description
0	Error
1	Successful



If this function is called, it remains active until the runtime system is terminated. That means every time the "strCondition" tag/expression becomes TRUE, the procedure is executed. If the tag/expression becomes TRUE, the function is not executed continuously, however, only once.

The procedure is executed on the client. To execute a procedure on the server, use the [RunGlobalProcedureOnServer, page 564](#) function.

Examples

If the value of the **TagOnTrue** variable becomes TRUE (not zero), the **UsingOnTrue** procedure is executed:

```
RunGlobalProcedureOnTrue( "TagOnTrue", "UsingOnTrue" )
```

19.14.28 RunVBScript

Executes a statement in VBScript language.

Function	Group	Execution	Windows	Embedded	Thin Client
RunVBScript	Module activity	synchronous	supported	supported	supported

Syntax

RunVBScript (strScript, "optTagReturnError")

Inputs**strScript**

Script statement that is to be executed by the function.

optTagReturnError

Name of the tag that is to receive the error (if any) generated by the statement (e.g., "Division by zero"). String type tags have to be put in quotation marks.

Appendix: WinStudio-Internal Scripting Language

Returned value

Value	Description
0	Error
1	Successful

Examples

Tag name	Expression
TagResult	RunVBScript("MsgBox(Time)") // Executes the MsgBox function page 744 from VBScript and displays the current time.
	RunVBScript(TagStatement) // Executes the statement configured in the value of the string tag TagStatement .
	RunVBScript("\$TagC=\$TagA/\$TagB", "TagError") // Writes in TagC the result of TagA divided by TagB . The error generated by the operation (if any) is written to the string tag TagError (e.g., "Division by zero").



This function is useful to execute VBScript statements in tasks that support the internal language only (e.g., scheduler groups). VBScript functions created in the global procedures can also be called.



The runtime station has to support the VBScript statements configured in this function in order to execute them.

19.14.29 SecureViewerReload

"SecureViewerReload" is an integrated scripting function that closes the "Secure Viewer" program and reloads it with a new configuration file.

Function	Group	Execution	Windows	Embedded	Thin Client
SecureViewerReload	Module activity	synchronous	not supported	not supported	Secure Viewer only

Syntax

```
SecureViewerReload(strFileName)
```

Inputs

strFileName

The file path of an INI file (*.ini) that describes the new configuration. (If the file is located in the same folder as **Viewer.exe**, only the file name is needed.) The file is to have the same structure and contain the same settings as the default configuration file (**Viewer.ini**).

This parameter can be a String tag or a constant text in quotation marks.

Returned value

This function does not return any value.

Examples

SecureViewerReload - Examples

```
SecureViewerReload( configFile1 )
```

```
SecureViewerReload( "C:\Program Files\Secure Viewer\Bin\Config1.ini" )
```

See also

- [Configuring the Thin Client, page 413](#)

19.14.30 SendKeyObject

Sends keyboard codes to the objects in the currently displayed WinStudio screen. This function can be used to trigger actions on dynamic screen objects.

Function	Group	Execution	Windows	Embedded	Thin Client
SendKeyObject	Module activity	synchronous	supported	not supported	not supported

This function cannot be used in combination with [tasks, page 57](#) or in the [global procedures, page 716](#) script.

Syntax

```
SendKeyObject( numEvent, strMainKey, numShift, numCtrl, numAlt, strTargetScreen, optNumID)
```

Inputs

numEvent

The event code for "On Down", "On While" or "On Up" of the [command animation, page 220](#) is as follows:

0	On Down
1	On While
2	On Up



When using the "On Down" event, the following is to be observed: If the function **SendKeyObject()** is called, WinStudio always sends the parameterized keyboard code exactly once. For this reason, all actions triggered by the keyboard code are started only once. This also applies for actions programmed in a screen object in the "On While" tab.

Appendix: WinStudio-Internal Scripting Language

strMainKey

Keyboard codes to be sent to WinStudio screens. The following values are permissible:

- "F1" ... "F20"
- "+"
- "-"
- "/"
- "**"
- "HOME"
- "END"
- "INSERT"
- "DELETE"
- "DOWN"
- "UP"
- "LEFT"
- "RIGHT"
- "PAGEUP"
- "PAGEDOWN"
- "SPACE"
- "RETURN"
- "BACKSPACE"
- "ESCAPE"
- "A" ... "Z"

numShift

A numeric value or tag defining whether the SHIFT key is pressed ("0": yes, "1": no).

numCtrl

A numeric value or tag defining whether the CTRL key is pressed ("0": yes, "1": no).

numAlt

A numeric value or tag defining whether the ALT key is pressed ("0": yes, "1": no).

strTargetScreen

The name of a WinStudio screen containing the keyboard code.



The "numShift", "numCtrl", "numAlt" and "strTargetScreen" parameters are optional. However, if one of the parameters is configured, the other parameters have to be configured as well.

Appendix: WinStudio-Internal Scripting Language

optNumID

The specific instance number of the screen. (The ID is assigned when the screen is opened with the [Open\(\)](#), [page 529](#) function.) This is an optional command parameter. The default ID is 0.

Returned value This function does not return any values.

Examples

Tag name	Expression
	SendKeyObject(0, "R", 1, 0, 0, "main", 10) // Sends Shift-R to the "main" screen with ID 10.

19.14.31 SetAppPath

Sets the new file path for the project folder. After this function is executed, "WinStudio" searches for all project files (i.e., screens, alarms, trends, database, events) in this folder.

Function	Group	Execution	Windows	Embedded	Thin Client
SetAppPath	Module activity	asynchronous	supported	supported	supported

Syntax

```
SetAppPath(strPath)
```

Inputs**strPath**

String tag or constant text in quotation marks with the file path

Returned value

0	Failed to set path
1	Succeeded in setting path

Examples

Tag name	Expression
	SetAppPath("C:\Studio\l")



If the computer is on a network, the **//IP address or host name/Path** syntax can be used to define a location on another network drive.

19.14.32 SetViewerInFocus

"SetViewerInFocus" is an internal scripting function that moves the viewer in front of all other open windows and maximizes it to fill the screen.

Function	Group	Execution	Windows	Embedded	Thin Client
SetViewerInFocus	Module activity	synchronous	supported	not supported	supported

Syntax

```
SetViewerInFocus()
```

There are no parameters.

Returned value This function does not return any value.

Appendix: WinStudio-Internal Scripting Language



With the introduction of Windows XP, system security functions prevent program windows from moving themselves in front of other windows without user input. If this function is called, the Windows taskbar indicates the activated window by blinking. (Anti-virus software may flag this mechanism as suspicious behavior.) The program window is not moved to the front before the program is selected by the user.

To work around this, **SetViewerInFocus** has to be called at least once in the [Startup Script, page 325](#) of the project. After the runtime system has booted completely and the function **SetViewerInFocus** has been called, the viewer is displayed in front.

Examples

SetViewerInFocus - Examples

```
SetViewerInFocus()
```

See also

- [Project Settings – Runtime Desktop tab, page 69](#)
- [Startup Script, page 325](#)

19.14.33 SetViewerPos

Sets the height, width and position of the project viewer or Thin Client.

Function	Group	Execution	Windows	Embedded	Thin Client
SetViewerPos	Module activity	synchronous	supported	not supported	supported

Syntax

```
SetViewerPos( numLeft, numTop, optNumWidth, optNumHeight )
```

Inputs

numLeft

A numeric tag that specifies the left-side position of the Viewer in pixels.

numTop

A numeric tag that specifies the top-side position of the Viewer in pixels.

optNumWidth

"Optional" numeric tag specifying the Viewer width in pixels.

optNumHeight

"Optional" numeric tag containing the Viewer height in pixels.

Returned value

0	Error
1	Successful

Examples

Tag name	Expression
Tag	SetViewerPos(50, 50, 640, 480)

The Viewer is opened with its default size set in the project settings if the optional parameters (**numWidth** and **numHeight**) are not specified.

Appendix: WinStudio-Internal Scripting Language

19.14.34 ShutDown

Closes all active modules of the runtime environment.

Function	Group	Execution	Windows	Embedded	Thin Client
ShutDown	Module activity	asynchronous	supported	supported	not supported

Syntax

```
ShutDown( )
```

This function does not have any parameters.

Returned value This function does not return any values.



This function does not close the <Development Environment>, [Database Spy, page 84](#) or [LogWin, page 85](#).

Examples

Tag name	Expression
	ShutDown()

19.14.35 StartTask

Starts a runtime module that is not currently running.

Function	Group	Execution	Windows	Embedded	Thin Client
StartTask	Module activity	asynchronous	supported	supported	executed on the server

Syntax

```
StartTask(strTask)
```

Inputs

strTask

The name of the task to be started (from the following list):

Value	Description
BGTask	Background tasks
Viewer	Viewer
DBSpy	Database Spy
LogWin	LogWin
Driver	Driver
UniDDECI	DDE client
UniNDDE	DDE server
UniODBC	ODBC
TCPServer	TCP/IP server
TCPClient	TCP/IP client
OPCClient	OPC client

Returned value This function does not return any values.

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
	StartTask("Viewer")



The **StartTask("Driver")** function starts all drivers configured in the project. To start a specific driver, the [WinExec\(\)](#), [page 574](#) function has to be used.

19.14.36 ViewerPostMessage

Sends a Windows system message to the specified WinStudio screen.

Function	Group	Execution	Windows	Embedded	Thin Client
ViewerPostMessage	Module activity	asynchronous	supported	supported	supported



This function cannot be used in combination with [tasks](#), [page 57](#) or in the [global procedures](#), [page 716](#) script.

Syntax

```
ViewerPostMessage( strScrTitle, numMessage, numwParam, numlParam, optNumID )
```

Inputs

strScrTitle

The name of the screen to which the message is to be sent.

numMessage

The number of the Windows system message to be sent.

numwParam

A numeric value or tag with a message-specific value, which is passed to the **wParam** parameter of the Windows system message.

numlParam

A numeric value or tag with a message-specific value, which is passed to the **lParam** parameter of the Windows system message.

optNumID

The specific instance number of the WinStudio screen. (The ID is assigned when the screen is opened with the [Open\(\)](#), [page 529](#) function.) This is an optional command parameter. The default ID is 0.

Returned value

This function does not return any values.



This function emulates the PostMessage function in Microsoft Windows. For further information including a complete list of available Windows system messages, please refer to the [Microsoft Developers Network](#).

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
	ViewerPostMessage("main", 16, 3, 1, 10) // Sends message 16 to the "main" screen with ID 10.

19.14.37 Wait

Pauses the execution of the runtime system for a specified number of milliseconds.

Function	Group	Execution	Windows	Embedded	Thin Client
Wait	Module activity	synchronous	supported	supported	supported

Syntax

```
Wait(numMillisec)
```

Inputs

numMillisec

A numeric variable or a constant value; the number of milliseconds for the waiting time.

Returned value

This function does not return any values.



This function is normally used only in [math worksheets, page 324](#), however, it can also be used in any [VBScript, page 715](#).

It is dangerous to use this function anywhere in the project as all incoming information from shared databases and connected devices is ignored as long as the runtime system is paused. Applications with this function have to be carefully tested.

Examples

Tag name	Expression
	Wait(1000)

19.14.38 WinExec

"WinExec" is an internal scripting function that executes a Windows command as if it was entered into the command prompt.

Function	Group	Execution	Windows	Embedded	Thin Client
WinExec	Module activity	asynchronous	supported	supported (see note)	supported

Syntax

```
WinExec(strCommand{ | , { optNumState | 0 | 1 | 2 | 3 | 4 | 7 } { | , { optNumSync | 0 | 1 } ,  
"optTagReturnOrHandle" } } )
```

Inputs

strCommand

String tag or constant text in quotation marks with the command for command execution

optNumState

Starting status of the program started by the command:

Appendix: WinStudio-Internal Scripting Language

Value	Description
0	Hides the program and transfers the control to another program
1	Activates and displays the program
2	Activates the program and displays it as an icon
3	Activates the program and maximizes it
4	Shows the program at the size it was last used The program is still active
7	Shows the program as an icon The program is still active

This parameter is optional. If no value is specified, the default value is "1".



This parameter is not supported on Windows CE operating systems. In this case, the transferred parameter value is ignored and the default value is used.

optNumSync

Parameter specifying whether the command is executed synchronously or asynchronously:

Value	Description
0	Asynchronous execution; the function immediately generates a value to be returned
1	Synchronous execution; the function creates a value to be returned as soon as the command has finished executing

This parameter is optional. If no value is specified, the default value is "0".



To verify that an asynchronously executed command has finished, use the "optTagReturnOrHandle" parameter as follows and the [WinExecIsRunning, page 576](#) function.

optTagReturnOrHandle

The name of a project tag that stores the returned value of the command:

- If the command is executed asynchronously, the tag receives a handle that can be used with the [WinExecIsRunning, page 576](#) function to determine whether the command is still running.
- If the command is executed synchronously, the tag receives the output code of the command. (This applies independent of the returned value of the function.)

This is an optional parameter, however, there is no default value.



The tag name has to be enclosed in quotation marks as shown in the syntax diagram. Otherwise, the project tries to call the value of the specified tag.

Returned value

This function returns the following possible values:

Appendix: WinStudio-Internal Scripting Language

Value	Description
0	Command was not executed successfully
1	Command was executed successfully

Please note that the returned value indicates only whether the execution of the command has been "started" successfully, particularly if it is executed asynchronously. It does not indicate when or how the command finished its execution.

Examples Starting Notepad and jumping immediately to the next row of the script:

WinExec("C:\Windows\System32\notepad.exe", 4)

Starting MS Paint and jumping immediately to the next row of the script:

WinExec("C:\Windows\System32\mspaint.exe")

Calling a batch file, executing it in hidden mode, waiting until it is finished and storing the output code in the tag "return":

WinExec("CMD /C call C:\Temp\MyBatch.bat", 0, 1, "return")

Calling a VBScript file, executing it in hidden mode, immediately continuing and storing the handle in the tag "handle":

WinExec("CMD /C call C:\Temp\MyScript.vbs", 0, 0, "handle")



Calling VBScript files is not supported on Windows CE target systems.

19.14.39 WinExecIsRunning

Checks whether a program started by the "WinExec" function is still running.

Function	Group	Execution	Windows	Embedded	Thin Client
WinExecIsRunning	Module activity	synchronous	supported	supported	supported

Syntax

```
WinExecIsRunning( numHandle, "optTagReturn" )
```

Inputs

numHandle

Handle number returned to the "optStrReturnorHandle" tag on calling the "WinExec" function.

"optTagReturn"

Tag with return value of the program that was called by means of the **WinExec** function.

Returned value

0	Successful execution
-1	Invalid parameter(s)
-2	Failed to open file Disk is write-protected or file name is invalid

Examples

Tag name	Expression
Tag	WinExecIsRunning(numHandle)
Tag	WinExecIsRunning(numHandle, "return")

19.15 File functions

19.15.1 File functions - General Information

These functions are used to read from, write to, print, move, and delete external files.

File

Function	Execution	Windows PC	Windows CE	Thin Client
DeleteOlderFiles, page 577	synchronous	supported	supported	supported
DirCreate, page 578	synchronous	supported	supported	supported
DirDelete, page 579	synchronous	supported	supported	supported
DirLength, page 579	synchronous	supported	supported	supported
DirRename, page 580	synchronous	supported	supported	supported
FileCopy, page 581	synchronous	supported	supported	supported
FileDelete, page 581	synchronous	supported	supported	supported
FileLength, page 582	synchronous	supported	supported	supported
FileRename, page 582	synchronous	supported	supported	supported
FindFile, page 584	synchronous	supported	supported	supported
FileWrite, page 583	synchronous	supported	supported	supported
FindPath, page 585	synchronous	supported	supported	supported
GetFileAttributes, page 585	synchronous	supported	supported	supported
GetFileTime, page 586	synchronous	supported	supported	supported
GetHstInfo, page 586	synchronous	supported	supported	supported
GetLine, page 587	synchronous	supported	supported	supported
HST2TXT, page 590	asynchronous	supported	supported	supported
HST2TXTIsRunning, page 591	synchronous	supported	not supported	executed on the server
PDFCreate, page 594	synchronous	supported	not supported	supported
Print, page 595	asynchronous	supported	supported	supported
RDFileN, page 595	synchronous	supported	supported	supported
WebGetFile, page 596	synchronous	supported	supported	supported

19.15.2 DeleteOlderFiles

Deletes files from a directory that are older than a specific date and correspond to the configured file mask.

Function	Group	Execution	Windows	Embedded	Thin Client
DeleteOlderFiles	Module activity	synchronous	supported	supported	supported

Syntax

```
DeleteOlderFiles( strPath, strMask, strDate )
```

Appendix: WinStudio-Internal Scripting Language

Inputs

strPath

The folder path to the files that are to be deleted.

strMask

The mask of the files to be deleted.

strDate

The cut-off date. All files older than this date are deleted.



The file format (e.g. **JMT** or **TMJ**) with the respective separator ("/", ":", ";", "." etc.) is specified in the project settings.

Returned value

Returns the number of files deleted.

Examples

Tag name	Expression
Tag	DeleteOlderFiles("C:\Studio\Project\HST\","*.hst", "04/12/2002")

19.15.3 DirCreate

Creates the specified directory.

Function	Group	Execution	Windows	Embedded	Thin Client
DirCreate	File	synchronous	supported	supported	executed on the server

Syntax

```
DirCreate( strDirectory, optBooFullPath )
```

Inputs

strDirectory

Path and name of the directory to be created.

optBooEmptyOnly

Optional parameter. If this parameter is not specified or if it is set to 0, the directory is only created if all parent directories exist. If this parameter has a value different from 0, the full path specified in the **strDirectory** parameter is created.

Returned value

Value	Description
-1	Invalid parameters
0	Failed to create the directory (e.g., drive does not exist)
1	Directory created successfully

Examples

Tag name	Expression
Tag	DirCreate("C:\Studio\Temp") // The Temp folder is created in the C:\Studio path (only if the C:\Studio path already exists).
Tag	DirCreate("C:\Studio\Temp",1) // The C:\Studio\Temp full path is created.

19.15.4 DirDelete

Deletes the specified directory.

Function	Group	Execution	Windows	Embedded	Thin Client
DirDelete	File	asynchronous	supported	supported	supported

Syntax

```
DirDelete( strDirectory, optBooEmptyOnly )
```

Inputs

strDirectory

Path and name of the folder to be deleted.



This parameter supports placeholders ("*" and "?").

optBooEmptyOnly

Optional function parameter. If this parameter has a value of 1, the directory is deleted only if it is empty. By default – i.e., if the parameter is not specified or has a value of 0 – the directory is deleted whether it is empty or not.

Returned value

Value	Description
-2	Attempted to delete a non-empty directory when this action is not allowed (i.e., optBooEmptyOnly does not equal 0)
-1	Invalid parameters
0	Failed to delete the directory (i.e., directory does not exist)
1	Directory deleted successfully

Examples

Tag name	Expression
Tag	DirDelete("C:\Studio\Temp") // The Temp folder from C:\Studio is deleted.
Tag	DirDelete("C:\Studio\Temp", 1) // The Temp folder from C:\Studio is deleted only if it is empty.

19.15.5 DirLength

Returns the size of a directory.

Function	Group	Execution	Windows	Embedded	Thin Client
DirLength	File	synchronous	supported	supported	executed on the server

Syntax

```
DirLength(strPath)
```

Inputs

strPath

Path of the directory to be checked.

Appendix: WinStudio-Internal Scripting Language

Returned value

-2	Directory does not exist
-1	Invalid parameters
n	Size (in bytes) of the files and subfolders in the directory

NOTICE

This function is executed synchronously. This means, the runtime system pauses while it waits for the function output.

If the specified directory contains large data amounts, this may lead to an interruption in processing of several seconds.

Examples

Tag name	Expression
Tag	DirLength("C:\Studio") // Returns the size (in bytes) of all files and subfolders from C:\Studio.

19.15.6 DirRename

Renames a directory.

Function	Group	Execution	Windows	Embedded	Thin Client
DirRename	File	asynchronous	supported	supported	executed on the server

Syntax

```
DirRename(strPath, strDirectoryFrom, strDirectoryTo)
```

Inputs

strPath

Path of the directory to be renamed



This function supports placeholders ("*" and "?").

strDirectoryFrom

The original name of the directory that is to be renamed.

strDirectoryTo

The new name of the directory.

Returned value

Value	Description
-1	Invalid parameters
0	Failed to rename the directory (e.g., strDirectoryFrom does not exist)
1	Directory renamed successfully.

Examples

Tag name	Expression
Tag	DirRename("C:\Studio\", "Temp", "New") // C:\Studio\Temp is renamed to C:\Studio\New.

19.15.7 FileCopy

Copies files to a higher directory

Function	Group	Execution	Windows	Embedded	Thin Client
FileCopy	File	synchronous	supported	supported	supported

Syntax

```
FileCopy(strSourceFile, strTargetFile{ | , optNumTimeOut})
```

Inputs

strSourceFile

Path and name of the file(s) to be copied.



This function supports placeholders ("*" and "?").

strTargetFile

The file path the file(s) are to be copied to.

optNumTimeOut

Numeric tag containing an integer to set the timeout period for the operation.

If this parameter is used, the function returns the value -1 after the specified timeout period.

The copying procedure is not cancelled by this. Instead, it creates an internal process to finish the copying procedure.

Returned value

Value	Description
-1	Timeout period expired.
0	Failed to copy file(s).
1	File(s) copied successfully.



This function is executed synchronously. This means, the project pauses while it waits for the function to complete. Large amounts of data and slow network connections may lead to longer program interruptions.

Examples

Tag name	Expression
Tag	FileCopy("C:\Studio\Project\HST*.hst", "C:\Temp\Hst\", 1000)
Tag	FileCopy("C:\Studio\Project\opert.txt", "C:\Temp\Tuesday_Report.txt", 500)

19.15.8 FileDelete

Deletes the specified file.

Function	Group	Execution	Windows	Embedded	Thin Client
FileDelete	File	synchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Syntax

FileDelete(strFile)

Inputs**strFile**

Path and name of the file to be deleted.

Returned value

Value	Description
0	Failed to delete file
Real	Returns the size of the deleted file

Examples

Tag name	Expression
Tag	FileDelete("C:\Studio\Readme.txt")

19.15.9 FileLength

Determines the size of a file.

Function	Group	Execution	Windows	Embedded	Thin Client
FileLength	File	synchronous	supported	supported	supported

Syntax

FileLength(strFile)

Inputs**strFile**

Path and name of the file.

Returned value

Returns the size of the specified file in bytes.

Examples

Tag name	Expression
Tag	FileLength("C:\Readme.txt")

19.15.10 FileRename

Renames a file.

Function	Group	Execution	Windows	Embedded	Thin Client
FileRename	File	synchronous	supported	supported	supported

Syntax

FileRename(strOldName, strNewName)

Inputs**strOldName**

Path and name of the file to be renamed.

strNewName

Path and new name of the file.

Returned value

This function does not return any value.

Appendix: WinStudio-Internal Scripting Language

Examples *FileRename - Examples*

```
FileRename( "C:\readme.txt", "C:\readthis.txt" )
```

19.15.11 FileWrite

Writes a string to a specified ASCII or Unicode file. If the file does not exist, it is created by the function.

Function	Group	Execution	Windows	Embedded	Thin Client
FileWrite	File	synchronous	supported	supported	supported

Syntax

```
FileWrite(strFileName, strWriteText{ | , optNumAppend })
```

Inputs

strFileName

Path and name of the file to be written with the function (String tag or string in quotation marks).

By default, the file is written to the project folder (i.e., the folder that contains the file **project_name.APP**). To write to another folder, enter the complete file path.

strWriteText

A string in quotation marks or a String tag containing the text to be written to the file.

optNumAppend

A numeric value or the name of the Integer tag with the ID for file handling (see table):

Value	Description
0	Creates a new ASCII file with the given file name. If the file already exists, it is overwritten.
1	Appends to an existing ASCII file with the given file name. If the file does not exist, a new file is created.
2	Creates a new Unicode file (UTF-16LE) with the specified file name. If the file already exists, it is overwritten.
3	Appends to an existing Unicode file with the given file name. If the file does not exist, a new file is created.

This parameter is optional. If any value other than 0 to 3 is specified or if the parameter is not used, the default value 0 is used.

Returned value

0	Successful execution
-1	Invalid parameter(s)
-2	Failed to open file Disk is write-protected or file name is invalid

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	FileWrite("c: est.txt", "This is a test")
Tag	FileWrite(strFileName, strWriteText)
Tag	FileWrite(strFileName, strWriteText, 1)

19.15.12 FindFile

Scripting function to search for files according to a specific keyword.

Function	Group	Execution	Windows	Embedded	Thin Client
FindFile	File	synchronous	supported	supported	supported

Syntax

```
FindFile(strFile{ | , "optTagFilesFound" { | , optNumTimeout } })
```

Inputs

strFile

Path and name of the file(s) to be searched for.

Placeholders ("*") can be used to find multiple files. For example, *.gif to find all GIF files or log*.txt to find all log files in a sequence (e.g., log001.txt, log002.txt, log003.txt).

By default, the function only searches the project folder. However, a file path (relative or absolute) can be specified to search a specific folder. For example: If **strFile** is defined as

\\drive name or IP address\Logs\log*.txt

...the function searches the **Logs** directory on the specified network drive.

optTagFilesFound

A tag array (of String type) that is to receive the names of the respective files. This array name has to be enclosed in quotation marks. Otherwise, the function tries to get the contents of the array.

This parameter is optional. If no value is specified, the file names are not saved and the function only returns the number of files found. For additional information, refer to "Returned value".



The array receives only the file names and "not" their paths, even if "strFile" is defined to search outside the default directory.

optNumTimeout

The timeout period (in milliseconds) after which the search function is cancelled.

This parameter is optional. If no value is specified, the function continues searching until the specified directory has been searched completely.

Returned value This function returns one of the following values:

Appendix: WinStudio-Internal Scripting Language

Value	Description
-1	Function timed out
0	No matching files found
n	Number of files found



This function may be called by any project client, however, it is always executed on the project server. By default, only the project folder of the server is searched. If "strFile" includes a relative file path, it has to be specified in relation to the project folder.

As the function is executed synchronously on the project server, an incorrect parameterization may lead to standstill of the server and all connected clients, particularly if the "optNumTimeout" parameter is not used.

Please note that the "optNumTimeout" parameter is not supported on Windows CE devices.

Examples Finding all text files in the project folder of the server:

FindFile("*.txt")

Finding all Microsoft Word files in the project folder and sending the file names to **StringArray** within a timeout period of 1000 milliseconds:

FindFile("*.doc", "StringArray", 1000)

19.15.13 FindPath

Checks whether a directory exists.

Function	Group	Execution	Windows	Embedded	Thin Client
FindPath	File	synchronous	supported	supported	supported

Syntax

FindPath(strPathName)

Inputs

strPathName

The file path to search for.

Returned value

Value	Description
0	Path not found
1	Path found

Examples

Tag name	Expression
Tag	FindPath("C:\Windows")

19.15.14 GetFileAttributes

Reads the attributes of a specified file.

Function	Group	Execution	Windows	Embedded	Thin Client
GetFileAttributes	File	synchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Syntax

```
GetFileAttributes(strFile)
```

Inputs**strFile**

Path and name of the file the attributes are to be determined of.

Returned value

Value	Description
-1	Error
1	Write-protected
2	Hidden
4	System
16	Directory
32	Archive
128	Normal
256	Temporary

Examples

Tag name	Expression
Tag	GetFileAttributes("C:\Readme.txt")

19.15.15 GetFileTime

Reads the time and date the file was last changed.

Function	Group	Execution	Windows	Embedded	Thin Client
GetFileTime	File	synchronous	supported	supported	supported

Syntax

```
GetFileTime(strFileName,numFormat)
```

Inputs**strFileName**

Path and name of the file.

numFormat

Parameter defining the format of the query:

- 0: Returns the date and time of the last file change.
- 1: Returns only the date of the last file change.
- 2: Returns only the time of the last file change.

Returned value

Returns the time and/or date of the last file change.

Examples

Tag name	Expression
Tag	GetFileTime("C:\Readme.txt")

19.15.16 GetHstInfo

Returns the start time, end time, and duration of the specified history (*.HST) file.

Appendix: WinStudio-Internal Scripting Language

Function	Group	Execution	Windows	Embedded	Thin Client
GetHstInfo	File	synchronous	supported	supported	supported

Syntax

GetHstInfo(strFileName, numInfoType)

Inputs

strFileName

Path and name of the history file to be read.

numFormat

Parameter specifying the type of information to be returned:

- 0: Returns the file recording start time.
- 1: Returns the file recording end time.
- 2: Returns the duration of file recording (in hours).

Returned value

Returns the start time, end time and recording duration in the history (*.HST) file. If the file cannot be read or the specified information cannot be returned, an error code is returned:

Value	Description
-1	Failed to retrieve the start time. Check whether the history file exists and is valid.
-2	Failed to retrieve the end time. Check whether the history file exists and is valid.
-3	Internal program error. Please contact technical support.
-4	The Studio TCP/IP server returned a time that is incompatible with the format specified in the project screen or web page. Please use the Verify Project, page 59 tool to update the project and try again.

Examples

Tag name	Expression
Tag	GetHstInfo("batch", 0)
Tag	GetHstInfo("hst/02060801.hst", 1)
Tag	GetHstInfo("C:\batch.bat", 2)

19.15.17 GetLine

Gets a specific line or search string from a text file and stores the line in a String tag.

Function	Group	Execution	Windows	Embedded	Thin Client
GetLine	File	synchronous	supported	supported	supported

Syntax

GetLine(strFileName, Search, "tagStore", optNumCase, "optOverflowTag", optRunFromServer)

Appendix: WinStudio-Internal Scripting Language

Inputs

strFileName

Path and name of the file to be searched. The parameter can be a fully qualified file path (e.g., **C:\File.txt**) or a simple file name (e.g., **File.txt**). In the latter case, the project searches for the file in the following directories:

- **Local Station:** The project searches for the file in the project folder and **Web** subfolder.
- **Thin Client:** If the parameter **optRunFromServer** is set to 0, the path where the file is to be searched is undetermined. If it is set to 1, the file is searched at the URL entered in the browser. If it is not found there, the search is continued under the backup URL.



For executing the function on a web client, it is recommended to set the **optRunFromServer** parameter to 1 and placing the files in the **Web** subfolder of the project.

Search

This parameter serves for addressing the line to be copied. There are two possibilities of line addressing depending on the file type of the parameter:

- If it is a **String** value or tag, the function searches the text file for the first occurrence of the string and copies the entire line containing the occurrence to the tag specified by **tagStore**. Additional occurrences are counted (see Returned Values below) but not copied.
- If it is a **numeric** value or tag, the function enters that line number in the text file and copies the line to the tag specified by **tagStore**. The first line of the file is line 0.

tagStore

Name of the String tag receiving the contents of the line pointed to by the function. This name has to be set in quotation marks. Otherwise, the function uses the content of the tag instead of its name.

optNumCase

"Optional" numeric tag specifying whether the search is case-sensitive.

- 0: Not case-sensitive
- 1: Case-sensitive

optOverflowTag

"Optional" numeric tag receiving the result of the overflow verification.

- 0: OK
- 1: Overflow

Appendix: WinStudio-Internal Scripting Language

optRunFromServer

"Optional" numeric tag that is only evaluated if the function is executed on a Thin Client. The parameter has the following meaning:

- 0: Retrieves the file from the Thin Client (do not use this value with names that are not fully qualified)
- 1: Retrieves the file from the web server. If the file name is not an URL, the function ignores the project path and searches for the file under the URL where the screen files are located.

Returned value If the function is successfully executed, it returns the total number of lines in which the search string was found. Otherwise, the function returns one of the following errors:

Value	Description
0	String was not found in the target file
-1	File not found
-2	Invalid "strFileName" parameter
-3	Invalid "search" parameter
-4	Invalid "tagStore" parameter
-5	Invalid "optNumCase" parameter
-6	Invalid "optOverflowTag" parameter
-7	Invalid number of parameters
-8	Invalid line number

Notes



This function only supports ASCII and UTF 16LE text encoding. (UTF 16LE is the Unicode implementation natively supported by Windows.) If this function is used to retrieve text from a UTF-8 or UTF-16BE encoded file, invalid characters may occur.



This function can only read up to 509 characters in one function call. If a line contains more than 509 characters (i.e., alphanumeric + CR + LF), the function interprets the string as several subsequent lines with the maximum length of 509 characters. The text from line 100 with 1024 characters (i.e., 1022 alphanumeric + CR + LF) is "virtually" distributed to lines 100, 101 and 102. For this reason, the function has to be called three times.

This affects all subsequent line accesses of the function. Line 101 of the source file is assigned the virtual line number 103, etc. For this reason, the source file is to be limited to 509 characters per line to prevent unnecessarily complicated counting of lines.

... to read line 100, the function has to be called three times:

```
GetLine( "C:\FileName.txt", 100, "strTagStore[1]") //Reads the first 509 characters
GetLine( "C:\FileName.txt", 101, "strTagStore[2]") //Reads the second 509 characters
GetLine( "C:\FileName.txt", 102, "strTagStore[3]") //Reads the last 6 characters
```

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	GetLine("C:\TechRef v61.doc", "Studio version 6.1", "ReturnedLine")// Gets the first occurrence of "Studio version 6.1".
Tag	GetLine("C:\Readme.txt", 1, "ReturnedLine", 0, "Overflow")// Gets the second line of the file.

19.15.18 HST2TXT

Exports information from a recorded trend file in binary format (*.hst) to a plain text (*.txt) or comma-separated (*.csv) file.

Function	Group	Execution	Windows	Embedded	Thin Client
HST2TXT	File	asynchronous	supported	supported	supported

Syntax

```
HST2TXT( strStartDate, strStartTime, numDuration, numGroup, strTargetFile, optStrSeparator,
         optNumMilliseconds, optStrFormat, optNumInterval )
```

Inputs

strStartDate

The start date of the data to be read.

strStartTime

The start time of the data to be read.

numDuration

Numeric tag with the duration of the data in hours.

numGroup

Numeric tag with the trend group number.

strTargetFile

Path and name of the file to be written.

optStrSeparator

"Optional" data separator character for the file. If this parameter is not specified, the function uses the TAB character (\t) to separate the values in the text file.

optNumMilliseconds

"Optional" numeric tag. If this parameter is false ("0"), the text file created does not show time stamps with millisecond-precision for each history sample.

Appendix: WinStudio-Internal Scripting Language

optStrFormat

String tag or string in quotation marks specifying the date format of entries in the target file:

- "DMY": Day, Month, Year
- "MDY": Month, Day, Year
- "YMD": Year, Month, Day

If this optional parameter is not specified, the function uses the format DMY for the time stamp in the text file.

optNumInterval

"Optional" numeric parameter specifying the sampling interval. Only row entries from this interval are written as text to the target file. All other row elements in the trend history are discarded.

For example, if **optNumInterval** has a value of 10, only every tenth row is written.

Returned value

Value	Description
-3	Invalid number of parameters
-2	Dll functions not found
-1	"InStudios.dll" not found
0	Function was executed successfully
1	Error. Previous execution of HST2TXT has not yet been completed

Examples

Tag name	Expression
Tag	HST2TXT("04/14/2002", "06:30:00", 0.1, 3, "C:\Studio\data.txt", "\"))
Tag	HST2TXT("04/14/2002", "06:30:00", 0.1, 3, "C:\Studio\data.csv", ",", "MDY")
Tag	HST2TXT("04/14/2002", "06:30:00", 0.1, 3, "C:\Studio\data.csv", ",", "MDY", 10)



If the comma character (,) is used as **optStrSeparator**, the function creates a file in CSV format (Comma Separated Values). This is useful for creating trend history files that can be opened with Microsoft Excel.

19.15.19 HST2TXTIsRunning

Returns the status of the "HST2TXT" function.

Function	Group	Execution	Windows	Embedded	Thin Client
HST2TXTIsRunning	File	synchronous	supported	not supported	executed on the server

Syntax

HST2TXTIsRunning()

This function does not have any parameters.

Appendix: WinStudio-Internal Scripting Language

Returned value

Value	Description
0	HST2TXT is still running
-1	Last conversion process was executed properly
-2	Reserved
-3	File not found. There are no history files in the configured time interval for the group specified
-4	Cannot open the HST file
-5	Cannot create/open the ASCII file
-6	Cannot read file information from the HST file
-7	Invalid file type
-8	Cannot read header information from the HST file
-9	Invalid number of tags in the header information ("0 > nTags > 250")
-10	Cannot create the Header file (.hdr)
-20	InStudiot.dll was not found
-30	Cannot access the dll function

Examples *HST2TXTIsRunning - Examples*

HST2TXTIsRunning()

19.15.20 LookupContains

This function checks whether an external file contains the specified keyword in its key column.

Function	Group	Execution	Windows	Embedded	Thin Client
LookupContains	File	synchronous	supported	supported	supported

Syntax

LookupContains(strKey)

Inputs**strKey**

The keyword to look for in the key column of the file.

Returned value

This function returns the following possible values:

Value	Description
0	Specified keyword not found
1	Specified keyword found



The external file has to be already loaded by calling the [LookupLoad, page 593](#) function.

Examples *LookupContains - Examples*

LookupContains("customer167")

19.15.21 LookupGet

This function calls a value from a table in an external file. The value in the values column is referenced by means of a key column.

Function	Group	Execution	Windows	Embedded	Thin Client
LookupGet	File	synchronous	supported	supported	supported

Syntax

```
LookupGet(strKey)
```

Inputs

strKey

The keyword to look for in the key column of the file.

Returned value

This function returns the value referenced by means of the keyword (as String).

If no value is found, this function returns **strKey**.



The external file has to be already loaded by calling the [LookupLoad, page 593](#) function.

Examples

LookupGet - Examples

```
LookupGet( "customer167" )
```

19.15.22 LookupLoad

This function loads an external text file that can be used to look up table values. One column of the file is designated as the keywords column and another column is designated as the values column.

Function	Group	Execution	Windows	Embedded	Thin Client
LookupLoad	File	synchronous	supported	supported	supported

Syntax

```
LookupLoad(strFileName,numColKey,numColValue,strDelimiters)
```

Inputs

strFileName

Path and name of the external file.

numColKey

The number of the column/field that contains the keywords.

numColValue

The number of the column/field that contains the desired values.

strDelimiters

The delimiter that separates the columns/fields.

Returned value

This function returns the number of rows in the specified file.

If the specified file cannot be found, this function returns a negative number as an error code.

Appendix: WinStudio-Internal Scripting Language



This function only loads the specified file. The file itself is not modified. To read the file, the [LookupContains, page 592](#) and [LookupGet, page 593](#) functions can be used.

Only one file can be loaded at a time. If the function is called repeatedly with a new file, the old file is deleted from the project memory.

Examples*LookupLoad - Examples*

```
LookupLoad( "C:\Temp\customerlist.csv", 1, 4, "", " ")
```

19.15.23 PDFCreate

Creates a PDF file from the specified source file.

Function	Group	Execution	Windows	Embedded	Thin Client
PDFCreate	File	synchronous	supported	not supported	supported

Syntax

```
PDFCreate(strSourceFile{ | ,optStrPdfFile })
```

Inputs**strSourceFile**

String specifying the file path and name of the desired source file (*.doc, *.txt, or *.rtf). If no complete path is specified, the function looks for the source file in the project folder.

optStrPdfFile

"Optional" string parameter specifying the file path and name of the created PDF file. If this parameter is not used, i.e., if no file path or name is specified, the PDF file is saved at the same location and with the same name as the source file. Only a new extension is added. For example:

\path\MyDocument.rtf becomes **\path\MyDocument.pdf**.



If the file name is entered without a path, a leading backslash ("\") is not necessary.

Returned value

Value	Description
0	Successful
1	Error in PDF profile information
3	Error saving PDF file
4	Job canceled
101	Error initializing PDF resource
102	Specified source file not found
103	Error generating PDF file
104	Incorrect number of parameters
105	Incorrect parameter type

Appendix: WinStudio-Internal Scripting Language



This function only supports the execution of one job at a time. If more than one user or command attempts to call the function at the same time, the function fails and returns a value of "101".

Examples

Tag name	Expression
	PDFCreate("C:\Report1.rtf")
	PDFCreate("C:\Report2.doc", "C:\Converted1.pdf")
	PDFCreate("C:\Report3.txt", "C:\Data\Converted1.pdf")

19.15.24 Print

Prints a text file.

Function	Group	Execution	Windows	Embedded	Thin Client
Print	File	asynchronous	supported	supported	supported

Syntax

```
Print(strFilePath{ | ,optNumOrientation })
```

Inputs

strFilePath

Path and name of the text file to be printed.

optNumOrientation

Parameter specifying the page orientation:

- 0 (default) = Portrait
- 1 = Landscape

Returned value No returned values.



The **optNumOrientation** parameter is not supported when running the project on a Windows CE target system.

Examples

Tag name	Expression
	Print("C:\ReadMe.txt")
	Print("C:\ReadMe.txt", 1)
	Print(TagFileName, 0)



This function can only be used to print text files. Data in any other format (e.g., pictures, binary files, etc.) cannot be printed with this function.

19.15.25 RDFileN

Launches a "File Browser" window to select a file.

Function	Group	Execution	Windows	Embedded	Thin Client
RDFileN	File	synchronous	supported	supported	supported

Syntax

```
RDFileN("tagSelectedFile", strSearchPath, strMask, optNumChangeDir)
```

Appendix: WinStudio-Internal Scripting Language

Inputs

tagSelectedFile

Parameter in form of a String tag in which the path and name of the file is entered that was selected in the file browser. The tag name **has to be** enclosed in quotation marks or the project tries to retrieve the contents of the tag. No VBScript variable name may be specified.

strSearchPath

The file path (directory) opened in the browser.

strMask

The mask to filter the files displayed in the browser.

optNumChangeDir

"Optional" numeric tag specifying whether the operator can change the browsing directory. If this parameter is not specified or set to "TRUE" (1), the open window enables to navigate to different directories. If it is set to "FALSE" (0), the window is restricted to the directory specified by **strSearchPath**.

Returned value

Value	Description
0	Successful
1	One of the parameters is not a string
2	Parameter 1 contains an invalid tag name
3	The user canceled the operation

Examples

Tag name	Expression
Tag	RDFileN("FileName", "C:\Studio\","*.doc", 1)

19.15.26 WebGetFile

Downloads a file from a specified web address and saves it locally.

Function	Group	Execution	Windows	Embedded	Thin Client
WebGetFile	File	synchronous	supported	supported	supported

Syntax

```
WebGetFile(strURL, strLocalPath)
```

Inputs

strURL

The URL (i.e. the web address) of the file to be downloaded (String tag or string in quotation marks).

strLocalPath

The full file path of where the file is to be saved (String tag or string in quotation marks).

Appendix: WinStudio-Internal Scripting Language

Returned value

Value	Description
-1	Timeout
0	File not found
1	File saved successfully

Examples

Tag name	Expression
Tag	WebGetFile("http://www.the-internet.com/myfile.txt", "C:\myfile.txt")
Tag	WebGetFile(myURL, myFilePath)

19.16 Graphic Functions

19.16.1 Graphic functions – General Information

These functions are used to handle and print WinStudio screens.

Graphic

Function	Execution	Windows PC	Windows CE	Thin Client
AutoFormat, page 597	synchronous	supported	supported	supported
GetScrInfo, page 598	synchronous	supported	supported	supported
PrintSetup, page 599	asynchronous	supported	supported	supported
PrintWindow, page 599	asynchronous	supported	supported	not supported
ResetDecimalPointsTable, page 601	synchronous	supported	supported	supported
RGBColor, page 601	synchronous	supported	supported	supported
RGBComponent, page 602	synchronous	supported	supported	supported
SetDecimalPoints, page 604	synchronous	supported	supported	supported
SetDisplayUnit, page 605	synchronous	supported	supported	supported
SetTagDisplayUnit, page 606	synchronous	supported	supported	supported

19.16.2 AutoFormat

Formats a real number with a specified number of decimal places according to the virtual formatting table created by the "SetDecimalPoints" function. (This is similar to the "Format" function, except that the number of decimal places does not need to be specified.)

Function	Group	Execution	Windows	Embedded	Thin Client
AutoFormat	Graphic	synchronous	supported	supported	supported

Syntax

AutoFormat(numValue)

Inputs

numValue

The real number to be formatted.

Returned value

This function returns a formatted string.

Appendix: WinStudio-Internal Scripting Language

Examples In the following examples, the [SetDecimalPoints, page 604](#) function has already been used to set 3 decimal places for values of at least 1.5 and 1 decimal place for values less than or equal to -3 before calling "AutoFormat".

Tag name	Expression
Tag	AutoFormat(1.543210) // Returned value = "1.543"
Tag	AutoFormat(-3.123456) // Returned value = "-3.1"

19.16.3 GetScrInfo

Gets information from the project about an open WinStudio screen.

Function	Group	Execution	Windows	Embedded	Thin Client
GetScrInfo	Graphic	synchronous	supported	supported	supported

Syntax

```
GetScrInfo(strScreenName, "tagResult", optNumResultType, optNumID)
```

Inputs

strScreenName

The name of the screen the information of which is to be retrieved.

tagResult

The name of the tag that is to receive the information returned by the function. This name has to be enclosed in quotation marks or the project tries to retrieve the contents of the tag. To return multiple values (optNumResultType = 0), an array tag has to be specified.

optNumResultType

Parameter for configuration of the return value

Value	Description
0	Default value. Writes the TOP, LEFT, BOTTOM and RIGHT screen coordinates to each consecutive position of the array tag, page 114 specified by the tagResult parameter
1	Writes the TOP screen coordinate to the tag specified by the tagResult parameter
2	Writes the LEFT screen coordinate to the tag specified by the tagResult parameter
3	Writes the BOTTOM screen coordinate to the tag specified by the tagResult parameter
4	Writes the RIGHT screen coordinate to the tag specified by the tagResult parameter

This parameter is optional. The default value is 0.

optNumID

The specific instance number of the screen. (The ID is assigned when the screen is opened with the [Open, page 529](#) function.) This parameter is optional. The default value is 0.

Appendix: WinStudio-Internal Scripting Language

Returned value

Value	Description
0	Successful
-1	The first and/or second parameters are no strings
-2	Memory allocation error
-3	optNumResultType is zero, but tagResult does not specify an array tag
-4	Invalid tag by the tagResult parameter

Examples

Tag name	Expression
TagErrorCode	GetScrInfo("main" , "TagXY[0]") // Retrieves the TOP, LEFT, BOTTOM and RIGHT coordinates of the "main" screen and then writes them to the first four positions of the array tag TagXY.
TagErrorCode	GetScrInfo("main", "TagXY", 3) // Retrieves the BOTTOM coordinate of the "main" screen and then writes it to TagXY.
TagErrorCode	GetScrInfo("main" , "TagXY", 2, 10) // Retrieves the LEFT coordinate of the "main" screen with ID 10 and then writes it to TagXY.

19.16.4 PrintSetup

Opens the standard configuration dialog of the operating system to select and configure the printer.

Function	Group	Execution	Windows	Embedded	Thin Client
PrintSetup	Graphic	asynchronous	supported	supported	supported



This function cannot be used in combination with [tasks, page 57](#) or in the [global procedures, page 716](#) script.

Syntax

PrintSetup()

This function does not have any parameters.

Returned value

No returned values.

Examples

PrintSetup - Examples

PrintSetup()

19.16.5 PrintWindow

Prints a screenshot of a WinStudio screen. The screen does not need to be open and active. The function can print a screen running in the background or even closed screen files.

Function	Group	Execution	Windows	Embedded	Thin Client
PrintWindow	Graphic	asynchronous	supported	supported	supported



This function cannot be used in combination with [tasks, page 57](#) or in the [global procedures, page 716](#) script.

Appendix: WinStudio-Internal Scripting Language

Syntax

PrintWindow(strScreenName , optNumOrientation, optNumID, optStrMnemonicList)

Inputs**strScreenName**

The name of the screen to be printed. If this parameter is not specified, the currently active screen is printed. (This parameter may not be specified if the function is executed on a Windows CE target system.)

optNumOrientation

Parameter specifying the paper orientation:

Value	Description
0	Portrait
1	Landscape

Default value is 0.



The **optNumOrientation** parameter is not supported when running the project on a Windows CE operating system.

optNumID

The specific instance number of the screen. (The ID is assigned when the screen is opened with the [Open, page 529](#) function.) This is an optional command parameter. The default ID is 0.

optStrMnemonicList

A string that describes how the custom properties of generic objects or [linked symbols, page 200](#) in the WinStudio screen are completed if the screen is printed. This string has the following syntax...

Label:Value

...where **Label** is the name of the property and **Value** is the tag, expression or literal value the property is to receive. Also more than one transfer value can be configured by separating them with space characters. Please refer to the following section with examples.



The **optStrMnemonicList** parameter does not work for a screen that is already open. If the screen has been opened, the custom properties have already received their default values.

Returned value

This function does not return any value.

Examples

Tag name	Expression
	PrintWindow() // Prints the currently active screen in portrait orientation.
	PrintWindow("Main", 1) // Prints the "Main" screen in landscape orientation.
	PrintWindow(TagScreenName) // Prints the screen specified by TagScreenName.

Appendix: WinStudio-Internal Scripting Language

Tag name	Expression
	PrintWindow("Main", 1, 10) // Prints the "Main" screen with ID 10.
	PrintWindow ("Main", 1, 0, "#Mne1:Tag1 #Mne2:Tag2") // Prints the "Main" screen, replacing the custom properties Mne1 and Mne2 with Tag1 and Tag2 , respectively.



This function can be used to print graphical reports that include [Alarm/Event Control, page 262](#) and [Trend Control, page 277](#) objects.

19.16.6 ResetDecimalPointsTable

Resets the virtual table of settings created by the "SetDecimalPoints" function.

Function	Group	Execution	Windows	Embedded	Thin Client
ResetDecimalPointsTable	Graphic	synchronous	supported	supported	supported

Syntax

```
ResetDecimalPointsTable( )
```

This function does not have any parameters.

Returned value

This function does not return any value.

Examples

Tag name	Expression
	ResetDecimalPointsTable() // Resets the virtual table of settings.

19.16.7 RGBColor

Returns the number of the color defined by the RGB (Red, Green, Blue) codes.

Function	Group	Execution	Windows	Embedded	Thin Client
RGBColor	Graphic	synchronous	supported	supported	supported

Syntax

```
RGBColor( numRed, numGreen, numBlue )
```

Inputs

numRed

RGB code for the red color component

numGreen

RGB code for the green color component

numBlue

RGB code for the blue color component

Returned value

This function returns the number of the color defined by the RGB (Red, Green, Blue) codes.

Examples

Appendix: WinStudio-Internal Scripting Language

Tag name	Expression
TagColor	RGBColor(51,153,102) // This function returns the value 13434828, which is the color code for Sea Green.
TagColor	RGBColor(TagRed,TagGreen,TagBlue) // This function returns the color code of the RGB values set in the tags TagRed, TagGreen and TagBlue, respectively.



A list of RGB codes and color values for the most used colors can be found in the [color interface, page 241](#) section.

19.16.8 RGBComponent

"RGBComponent" is an internal scripting function determining the color code of one of the three basic colors (red, green or blue) from a color code.

Function	Group	Execution	Windows	Embedded	Thin Client
RGBComponent	Graphic	synchronous	supported	supported	supported

Syntax

```
RGBComponent(numColor, { numComponent | 0 | 1 | 2 })
```

Inputs

numColor

The decimal code for a 24-bit RGB color that can be any integer value between 0 and 16777215. (This color model is also referred to as "Truecolor" or "16 millions of colors".)

numComponent

The basic color to be determined: "0" is red, "1" is green and "2" is blue.

Returned value

This function returns an integer value between 0 and 255 representing the respective color code of the configured basic color.



For a list of frequently used RGB color codes and their equivalent "plain English" names, refer to [Color interface, page 241](#).

Examples

Retrieving the red basic color in color code 13434828 (i.e., sea green):

```
RGBComponent( 13434828, 0 )
```

Retrieving the RGB code for the basic color (red, blue, green) specified in **TagComponent** from the color code specified in **TagCode**.

```
RGBComponent( TagCode, TagComponent )
```

19.16.9 SaveScreenShot

This function takes a screen shot of the WinStudio screen and saves it to an image file.

Function	Group	Execution	Windows	Embedded	Thin Client
SaveScreenShot	Graphic	synchronous	supported	supported (only for open screens)	executed on the server

Appendix: WinStudio-Internal Scripting Language

Syntax

```
SaveScreenShot({ | optStrScreenName{ | ,optStrOutputFile{ | ,{ optNumFormat | 0 | 1 | 2 | 3 | 4 | 5 } } } })
SaveScreenShot(
SaveScreenShot( optStrScreenName )
SaveScreenShot( optStrScreenName, optStrOutputFile )
SaveScreenShot( optStrScreenName, optStrOutputFile, optNumFormat )
```

Inputs

optStrScreenName

The name of a screen file (*.scr). If no file path is specified, the file has to be located in the "Screens" folder under the project folder (e.g., \projectname\Screen\screenname.scr).

If the screenshot function is executed on a Windows PC, the screens do not need to be open. On Windows CE devices, however, printing is only possible if the screens are open.

This is an optional parameter. If no value (or "") is specified, the currently open and active screen is used.

optStrOutputFile

The name of the output file. If no file path is specified, the screen file is saved in the Web folder of the project (e.g., \projectname\Web\screenname.jpg)

This is an optional parameter. If no value is specified, the screen name of the selected screen (optStrScreenName) is used.

optNumFormat

The format of the screen file:

Value	Description
0	BMP
1	JPG
2	PNG
3	GIF
4	TIFF
5	Auto

This is an optional parameter. If no value is specified, the default is "1" (JPG).

Returned value

This function returns the following possible values:

Value	Description
0	Successful
-1	Incorrect number of parameters
-2	Incorrect parameter types
-3	Invalid directory
-4	Second parameter may not be empty
-5	Incorrect format / invalid option for third parameter
-6	Failed to save file

Appendix: WinStudio-Internal Scripting Language

Value	Description
-7	Failed to create compatible bitmap
-10000	Project is not running. (This typically occurs when trying to use the function in the Database Spy.) See note



For this function to be properly executed, the project has to be running and the Viewer task has to be started.

Examples Saving the currently active screen to \projectname\Web\ScreenShot.JPG:

SaveScreenShot()

Saving the screen file **main.scr** to \projectname\Web\main.JPG:

SaveScreenShot("main.scr")

Saving the currently active screen as a BMP with the name of the currently logged-on user:

SaveScreenShot("", UserName, 0)

19.16.10 SetDecimalPoints

Sets the number of decimal places to be displayed, for a specified range of real numbers. This setting is to be used by all input and dynamic objects that have the **Auto Format** option enabled as well as by the **AutoFormat** function itself.

Function	Group	Execution	Windows	Embedded	Thin Client
SetDecimalPoints	Graphic	synchronous	supported	supported	supported

Syntax

SetDecimalPoints(numBaseValue, numDecimalPoints)

Inputs

numBaseValue

The base value of the range of real numbers. For negative values, the range includes all real numbers less than or equal to that value. For positive values, the range includes all real numbers greater than or equal to that number. (To specify both value ranges, the function has to be called twice with the respective values)

numDecimalPoints

The number of decimal places to be displayed for the range of real numbers specified by "numBaseValue".

Returned value

Value	Description
0	Error
1	Successful



If this function is called repeatedly with different parameters, a virtual table with format settings can be created. A different number of decimal places can be set for each range of real numbers and all of the settings are saved for the duration of runtime or until the table is reset using the [ResetDecimalPointsTable, page 601](#) function.

Appendix: WinStudio-Internal Scripting Language



This formatting does "not" change the actual value of any tag or expression. It only changes how the values of screen objects are displayed.

Examples

Tag name	Expression
Tag	SetDecimalPoints(1.5, 3) // Displays 3 decimal places for all real numbers greater than or equal to 1.5.
Tag	SetDecimalPoints(-3, 1) // Displays 1 decimal place for all real numbers less than or equal to -3.

19.16.11 SetDisplayUnit

Searches for all tags and all Grid object and Trend Control object values with a specific engineering unit (refer to the **Unit** tag field) and sets the **DisplayUnit**, **UnitDiv** and **UnitAdd** fields for these tags.

Function	Group	Execution	Windows	Embedded	Thin Client
SetDisplayUnit	Graphic	synchronous	supported	supported	supported

Syntax

```
SetDisplayUnit(strUnitOrigin, strDisplayUnit, numDiv, numAdd)
```

Inputs

strUnitOrigin

The original unit.

strDisplayUnit

The new value for the **DisplayUnit** tag field.

numDiv

The new value for the **UnitDiv** tag field.

numAdd

The new value for the **UnitAdd** tag field.

Returned value

Value	Description
0	Successful
-1	Incorrect number of parameters
-2	"strUnitOrigin" parameter is empty
-3	"numDiv" parameter is invalid (equal to 0)



This function only affects how the tags are displayed. The actual tag values are not changed in any way.

Examples

Appendix: WinStudio-Internal Scripting Language

Tag name	Expression
Tag	SetDisplayUnit("C", "F", 0.555556, 32) // For all tags and object values with a Unit of "C", the DisplayUnit tag field is set to "F", the UnitDiv tag field is set to 0.555556, and the UnitAdd tag field is set to 32.

19.16.12 SetTagDisplayUnit

Sets the **DisplayUnit**, **UnitDiv** and **UnitAdd** variable properties for a specific variable.

Function	Group	Execution	Windows	Embedded	Thin Client
SetTagDisplayUnit	Graphic	synchronous	supported	supported	supported

Syntax

```
SetTagDisplayUnit( strTagName, strDisplayUnit, numDiv, numAdd )
```

Inputs

strTagName

Name of the variable whose **DisplayUnit**, **UnitDiv** and **UnitAdd** properties are to be set. If a String variable is assigned to this parameter, it is to "include" the name of the variable whose properties are to be set.

strDisplayUnit

The new value for the **DisplayUnit** tag property.

numDiv

The new value for the **UnitDiv** tag property.

numAdd

The new value for the **UnitAdd** tag property.

Returned value

Value	Description
0	Successful
-1	Incorrect number of parameters
-2	Specified tag does not exist
-3	numDiv parameter is invalid (equal to 0)

Examples

Tag name	Expression
Tag	SetTagDisplayUnit("TagTemp", "F", 0.555556, 32) // For the tag "TagTemp", the DisplayUnit tag field is set to "F", the UnitDiv tag field is set to 0.555556, and the UnitAdd tag field is set to 32.

19.17 Translation Functions

19.17.1 Translation Functions – General Information

These functions serve for translation of texts during runtime.

Appendix: WinStudio-Internal Scripting Language

Translation

Function	Execution	Windows PC	Windows CE	Thin Client
Ext, page 607	synchronous	supported	supported	supported
SetLanguage, page 607	synchronous	supported	supported	supported
SetTranslationFile, page 608	synchronous	supported	supported	supported

19.17.2 Ext

Translates the specified text by means of the active translation file (refer to "SetLanguage" or "SetTranslationFile").

Function	Group	Execution	Windows	Embedded	Thin Client
Ext	Translation	synchronous	supported	supported	supported

Syntax

```
Ext(strText)
```

Inputs

strText

The text to be translated.

Returned value

Returns the text translation using the active translation file.

Examples

Tag name	Expression
Tag	Ext("Start") // Returned value if translating to Portuguese = "Iniciar"
Tag	Ext("Stop") // Returned value if translating to German = "Stopp"

19.17.3 SetLanguage

This function changes the language of the project to one of the translations configured in the translation table.

Function	Group	Execution	Windows	Embedded	Thin Client
SetLanguage	Translation	synchronous	supported	supported	supported

Syntax

```
SetLanguage( numLanguageCode )
```

Inputs

numLanguageCode

The code for the language to be activated. The language has to be already configured in the [translation table, page 381](#).

Returned value

This function returns the following possible values:

Value	Description
0	Error
1	Successful



The language codes are defined by [ISO 639-1](#).

Appendix: WinStudio-Internal Scripting Language

Examples Setting the language to "English – United States":

SetLanguage(1033)

Setting the language to "French – France":

SetLanguage(1036)

19.17.4 SetTranslationFile

Sets the active translation file and translates all enabled texts within the project.

Function	Group	Execution	Windows	Embedded	Thin Client
SetTranslationFile	Translation	synchronous	supported	supported	supported

Syntax

```
SetTranslationFile(strFileName{ | ,optStrColumnName })
```

Inputs

strFileName

The name of the translation file

optStrColumnName

The name of the column in the translation file the texts of which are to be used for translation. This parameter is optional. By default, the second column of the translation file is used.

Returned value

Value	Description
0	Successful
-1	Invalid number of parameters
-2	Incorrect parameter type
-3	Translation file could not be found or opened

Examples

Tag name	Expression
Tag	SetTranslationFile("Portuguese.tra")
Tag	SetTranslationFile("German.tra")
Tag	SetTranslationFile("Mytranslation.csv" , "Portuguese")
Tag	SetTranslationFile("Mytranslation.csv" , "German")



For this function to work correctly, the translation option from the [Project Settings, page 62](#) dialog have to be set.

NOTICE

A translation file has to be available in the [Translation tool, page 381](#).

19.18 Multimedia Functions

19.18.1 Multimedia Functions – General Information

These functions are used to play external audio and video files.

19.18.2 Play

Plays a specified WAV audio file.

Function	Group	Execution	Windows	Embedded	Thin Client
Play	Multimedia	synchronous	supported	supported	supported



For this function to work on a Thin Client, the target WAV file has to be located in the same file path as on the remote station.

Syntax

```
Play(strFileName{ | ,optNumSynchronous })
```

Inputs

strFileName

The file path and name of the WAV file to be played.

optNumSynchronous

A numeric parameter specifying whether the function is to be executed synchronously or asynchronously:

Value	Description
0	Asynchronous (i.e., the runtime environment continues without waiting for the function to complete)
1	Synchronous (i.e., the runtime environment continues and waits for the function to complete)

This is an optional parameter. If no value is specified, the default value is "0".

Returned value

This function does not return any value.

Examples

Tag name	Expression
	Play("C:\Sounds\Wav\alarm.wav")

19.19 System Information Functions

19.19.1 System Information Functions – General Information

These functions are used to receive information about the computer that is running the project (either server or client, depending on the function) as well as to change various project settings on that computer.

System info

Function	Execution	Windows PC	Windows CE	Thin Client
DbVersion, page 611	synchronous	supported	supported	supported
GetAppHorizontalResolution, page 611	synchronous	supported	not supported	executed on the server
GetAppPath, page 612	synchronous	supported	supported	executed on the server
GetAppVerticalResolution, page 612	synchronous	supported	not supported	executed on the server

Appendix: WinStudio-Internal Scripting Language

Function	Execution	Windows PC	Windows CE	Thin Client
GetComputerIP, page 612	synchronous	supported	supported	supported
GetComputerName, page 613	synchronous	supported	not supported	supported
GetCursorX, page 613	synchronous	supported	supported	supported
GetCursorY, page 613	synchronous	supported	supported	supported
GetDisplayHorizontalResolution, page 614	synchronous	supported	supported	supported
GetDisplayVerticalResolution, page 614	synchronous	supported	supported	supported
GetFreeMemoryCE, page 614	synchronous	not supported	supported	supported
GetHardKeyModel, page 615	synchronous	supported	not supported	executed on the server
GetHardKeySN, page 615	synchronous	supported	not supported	executed on the server
GetIPAll, page 616	synchronous	supported	supported	supported
GetMemoryCE, page 616	synchronous	not supported	supported	supported
GetNetMACID, page 617	synchronous	supported	supported	supported
GetOS, page 618	synchronous	supported	supported	supported
GetPrivateProfileString, page 618	synchronous	supported	supported	supported
GetProductPath, page 619	synchronous	supported	supported	supported
GetRegValue, page 619	synchronous	supported	supported	not supported
GetRegValueType, page 620	synchronous	supported	supported	not supported
GetServerHostName, page 621	synchronous	not supported	not supported	supported
GetTickCount, page 621	synchronous	supported	supported	supported
InfoAppAlrDir, page 622	synchronous	supported	supported	supported
InfoAppHSTDir, page 622	synchronous	supported	supported	supported
InfoDiskFree, page 622	synchronous	supported	not supported	supported
InfoResources, page 623	synchronous	supported	supported	supported
IsActiveXReg, page 623	synchronous	supported	supported	supported
IsAppChangedOnServer, page 624	synchronous	supported	supported	supported
NoInputTime, page 625	synchronous	supported	supported	supported
ProductVersion, page 625	synchronous	supported	supported	supported
RegSaveCE, page 626	synchronous	not supported	supported	not supported
ReloadAppFromServer, page 626	synchronous	supported	supported	supported
SaveAlarmFile, page 627	synchronous	supported	supported	not supported
SetAppAlarmPath, page 628	synchronous	supported	supported	executed on the server
SetAppHSTPath, page 628	synchronous	supported	supported	executed on the server
SetDateFormat, page 629	synchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Function	Execution	Windows PC	Windows CE	Thin Client
SetKeyboardLanguage, page 629	synchronous	supported	supported	supported
SetRegValue, page 630	synchronous	supported	supported	not supported
SetWebConfig, page 631	synchronous	supported	supported	executed on the server
SNMPGet, page 633	synchronous	supported	not supported	supported
SNMPSet, page 634	synchronous	supported	not supported	supported
WritePrivateProfileString, page 636	synchronous	supported	supported	supported

19.19.2 DbVersion

"DbVersion" is an internal scripting function that retrieves the version number of the project tags database.

Function	Group	Execution	Windows	Embedded	Thin Client
DbVersion	System info	synchronous	supported	supported	supported

Syntax

```
DbVersion()
```

This function does not have any parameters.

Returned value This function returns a numerical value equal to the version number of the tags database.



This function only applies to the native database within the project. Currently, there is no function to retrieve the version number of an external or historical database.

Examples *DbVersion - Examples*

```
DbVersion()
```

19.19.3 GetAppHorizontalResolution

Gets the default horizontal screen resolution (in pixels) of the project.

Function	Group	Execution	Windows	Embedded	Thin Client
GetAppHorizontalResolution	System info	synchronous	supported	not supported	executed on the server

Syntax

```
GetAppHorizontalResolution()
```

This function does not have any parameters.

Returned value Returns the default value as specified in the project configuration. The Windows configuration is not taken into account.

Examples

Tag name	Expression
Tag	GetAppHorizontalResolution() // Returned value = 640
Tag	GetAppHorizontalResolution() // Returned value = 800

Appendix: WinStudio-Internal Scripting Language

19.19.4 GetAppPath

Returns the file path of the project folder.

Function	Group	Execution	Windows	Embedded	Thin Client
GetAppPath	System info	synchronous	supported	supported	executed on the server

Syntax

GetAppPath()

This function does not have any parameters.

Returned value Returns the file path as a string.**Examples**

Tag name	Expression
Tag	GetAppPath() // Returned value = "C:\DemoProject\"
Tag	GetAppPath() // Returned value = "C:\Studio\Projects\project_name\"



This function always returns a "\" at the end of the path.

19.19.5 GetAppVerticalResolution

Gets the default vertical screen resolution (in pixels) of the project.

Function	Group	Execution	Windows	Embedded	Thin Client
Avg	System info	synchronous	supported	supported	supported

Syntax

GetAppVerticalResolution()

This function does not have any parameters.

Returned value Returns the default value as specified in the project configuration. The Windows configuration is not taken into account.**Examples**

Tag name	Expression
Tag	GetAppVerticalResolution() // Returned value = 480
Tag	GetAppVerticalResolution() // Returned value = 600

19.19.6 GetComputerIP

Returns the first IP address of the local computer.

Function	Group	Execution	Windows	Embedded	Thin Client
GetComputerIP	System info	synchronous	supported	supported	supported

Syntax

GetComputerIP()

This function does not have any parameters.

Returned value Returns the first IP address of the local station as a string.

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	GetComputerIP() // Returned value = "192.168.0.1"
Tag	GetComputerIP() // Returned value = "248.12.2.78"

19.19.7 GetComputerName

Returns the local computer name.

Function	Group	Execution	Windows	Embedded	Thin Client
GetComputer-Name	System info	synchronous	supported	not supported	supported

Syntax

```
GetComputerName()
```

This function does not have any parameters.

Returned value

Returns the local computer name as a string.

Examples

Tag name	Expression
Tag	GetComputerName() // Returned value = "Terminal53"
Tag	GetComputerName() // Returned value = "BobsComputer"

19.19.8 GetCursorX

Gets the X-coordinate of the cursor on the screen.

Function	Group	Execution	Windows	Embedded	Thin Client
GetCursorX	System info	synchronous	supported	supported	supported

Syntax

```
GetCursorX()
```

This function does not have any parameters.

Returned value

This function returns the X-coordinate of the cursor on the screen or "-1" if an error occurs.

Examples

Tag name	Expression
	GetCursorX() // Returned value = 1024

19.19.9 GetCursorY

Gets the Y-coordinate of the cursor on the screen.

Function	Group	Execution	Windows	Embedded	Thin Client
GetCursorY	System info	synchronous	supported	supported	supported

Syntax

```
GetCursorY()
```

This function does not have any parameters.

Returned value

This function returns the Y-coordinate of the cursor on the screen or "-1" if an error occurs.

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
	GetCursorY() // Returned value = 768

19.19.10 GetDisplayHorizontalResolution

Gets the horizontal resolution (in pixels) of the display connected to the local station.

Function	Group	Execution	Windows	Embedded	Thin Client
GetDisplayHorizontalResolution	System info	synchronous	supported	supported	supported

Syntax

```
GetDisplayHorizontalResolution()
```

This function does not have any parameters.

Returned value

This function returns the horizontal resolution of the display as an integer.

Examples

Tag name	Expression
	GetDisplayHorizontalResolution() // Returned value = 1024

19.19.11 GetDisplayVerticalResolution

Gets the vertical resolution (in pixels) of the display connected to the local station.

Function	Group	Execution	Windows	Embedded	Thin Client
GetDisplayVerticalResolution	System info	synchronous	supported	supported	supported

Syntax

```
GetDisplayVerticalResolution()
```

This function does not have any parameters.

Returned value

This function returns the vertical resolution of the display as an integer.

Examples

Tag name	Expression
	GetDisplayVerticalResolution() // Returned value = 768

19.19.12 GetFreeMemoryCE

Returns the free memory available on a Windows Embedded device.

Function	Group	Execution	Windows	Embedded	Thin Client
GetFreeMemoryCE	System info	synchronous	not supported	supported	supported

Syntax

```
GetFreeMemoryCE({ | optNumType })
```

Appendix: WinStudio-Internal Scripting Language

Inputs

optNumType

A numeric flag that specifies which type of free memory the project is to retrieve from the device:

0 – Total free program memory

This is an optional parameter. If no value is specified, the default value is "0".

Returned value

Value	Description
>0	Size of free memory in bytes
-1	CoreDll.dll file not found
-2	GetMemoryCE , page 616 function not found
-3	Invalid optional parameter
-4	Type of memory unavailable

Examples

Tag name	Expression
Tag	GetFreeMemoryCE()

19.19.13 GetHardKeyModel

Returns the model name of the Hardkey.

Function	Group	Execution	Windows	Embedded	Thin Client
GetHardKeyModel	System info	synchronous	supported	not supported	executed on the server

Syntax

```
GetHardKeyModel()
```

This function does not have any parameters.

Returned value

If the Hardkey is installed, the function returns a string with the hardkey model name.

If the hardkey is not installed, the function returns "0".



The hardkey has to be connected before executing this function. Otherwise, the function does not execute properly.

Examples

Tag name	Expression
Tag	GetHardKeyModel() // Returned value = "Local Interface"
Tag	GetHardKeyModel() // Returned value = "Advanced Server"

19.19.14 GetHardKeySN

Returns the serial number of the hardkey.

Function	Group	Execution	Windows	Embedded	Thin Client
GetHardKeySN	System info	synchronous	supported	not supported	executed on the server

Syntax

```
GetHardkeySN()
```

Appendix: WinStudio-Internal Scripting Language

This function does not have any parameters.

Returned value If the hardkey is installed, the function returns a string with the serial number of the hardkey.
If the hardkey is not installed, the function returns "0".



The hardkey has to be connected before executing this function. Otherwise, the function does not execute properly.

Examples

Tag name	Expression
Tag	GetHardkeySN() // Returned value = 120.745
Tag	GetHardkeySN() // Returned value = 224.941

19.19.15 GetIPAll

Returns all IP addresses of the local station and stores them in a String array tag.

Function	Group	Execution	Windows	Embedded	Thin Client
GetIPAll	System info	synchronous	supported	supported	supported

Syntax

```
GetIPAll( "tagArrayIP", optRefresh )
```

Inputs**tagArrayIP**

Name of the String array tag to receive the IP addresses. This name has to be enclosed in quotation marks or the project tries to get the contents of the array tag.

optRefresh

"Optional" tag triggering an update of this function. Every time this tag changes its value, the project updates the value returned by the function.

Returned value

Value	Description
"n"	Number of IP addresses found
-1	Invalid number of parameters
-2	Invalid parameter type

Examples

Tag name	Expression
Tag	GetIPAll("TagArrayIP") // Returned value = 1
Tag	GetIPAll("TagArrayIP", Second) // Returned value = 2

19.19.16 GetMemoryCE

Returns the total memory available in a Windows CE device.

Function	Group	Execution	Windows	Embedded	Thin Client
GetMemoryCE	System info	synchronous	not supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Syntax

```
GetMemoryCE({ | optNumType })
```

Inputs

optNumType

Numeric parameter to specify the type of memory to be determined:

Value	Description
0	Total program memory
1	Total storage memory
2	Total memory

This is an optional parameter. If no value is specified, the default value is "0".

Returned value

Value	Description
>0	Size of memory in bytes
-1	Coredll.dll file not found
-2	GetMemoryCE function not found
-3	Invalid optional parameter

Examples

Tag name	Expression
Tag	GetMemoryCE(1)

19.19.17 GetNetMACID

Gets the MAC address of the installed network adapters.

Function	Group	Execution	Windows	Embedded	Thin Client
GetNetMACID	System info	synchronous	supported	supported	supported

Syntax

```
GetNetMACID( "optTagMACID", "optTagAdapterName" )
```

Inputs

optStrMACID

Name of the String tag in which the MAC address of the network adapter is to be stored. If more than one network adapter is installed on the station, a String type array tag can be configured for this parameter.

optStrAdapterName

Name of the String tag in which the name of the network adapter is to be stored. If more than one network adapter is installed on the station, a String type array tag can be configured for this parameter.

Returned value

Value	Description
>0	Number of network adapters found
0	No network adapters found
-1	Invalid number of parameters

Appendix: WinStudio-Internal Scripting Language

Value	Description
-2	One of the parameters is not a string
-3	Tag configured in optTagMACID does not exist
-4	Tag configured in optTagAdapterName does not exist

Examples

Tag name	Expression
NumNIC	GetNetMACID("MACIDTag")
NumNIC	GetNetMACID("MACIDTag", "AdapterName")
NumNIC	GetNetMACID("MACIDTag[1]", "AdapterName[1]")

19.19.18 GetOS

Gets the current operating system.

Function	Group	Execution	Windows	Embedded	Thin Client
GetOS	System info	synchronous	supported	supported	supported

Syntax

```
GetOS()
```

This function does not have any parameters.

Returned value

Value	Description
0	Windows 3.11
1	Windows 95/98/ME
2	Windows 2000/XP/Vista/7
3	Windows CE/Mobile

Examples

Tag name	Expression
Tag	GetOS() //Returned value = 2

19.19.19 GetPrivateProfileString

Reads a specified parameter from an .ini file using the standard .ini format.

Function	Group	Execution	Windows	Embedded	Thin Client
GetPrivateProfileString	System info	synchronous	supported	supported	supported

Syntax

```
GetPrivateProfileString( strSection, strName, strDefault, strFileName )
```

Inputs

strSection

The section name of the ini file to be read.

strName

The parameter of the ini file to be read.

Appendix: WinStudio-Internal Scripting Language

strDefault

The default setting for the "strName" parameter to be read. If the parameter is not found in the .ini file, the function returns the default value specified here.

strFileName

The path and name of the .ini file to be read.

Returned value Returns the value of the specified parameter.

Examples

Tag name	Expression
Tag	GetPrivateProfileString("boot loader", "timeout", "50", "C:\boot.ini") // Returned value = 30

19.19.20 GetProductPath

Gets the path to the program directory.

Function	Group	Execution	Windows	Embedded	Thin Client
GetProductPath	System info	synchronous	supported	supported	supported

Syntax

```
GetProductPath( )
```

This function does not have any parameters.

Returned value Returns the path to the program directory as a string.

Examples

Tag name	Expression
Tag	GetProductPath() // Returned value = "C:\Program Files\Rexroth\IndraWorks\WinStudio"

19.19.21 GetRegValue

Gets the value of a registry entry in the Windows registry.

Function	Group	Execution	Windows	Embedded	Thin Client
GetRegValue	System info	synchronous	supported	supported	not supported

Syntax

```
GetRegValue( numMainKey, strKey, strValueName )
```

Inputs

numMainKey

Numeric tag to select the main key (see table):

Value	Description
0	HKEY_LOCAL_MACHINE
1	HKEY_CLASSES_ROOT
2	HKEY_CURRENT_USER
3	HKEY_USERS

Appendix: WinStudio-Internal Scripting Language

Value	Description
4	HKEY_CURRENT_CONFIG
5	HKEY_PERFORMANCE_DATA

strKey

Path for the registry entry to be read under the main key (String variable or string in quotation marks)

strVariableName

Name of the registry entry to be retrieved. The maximum length is 255 characters.

Returned value If the function succeeds, the value of the registry entry is returned. Otherwise one of the following error codes is returned:

Value	Description
-1	Invalid number of parameters or invalid main key
-2	Variable type is not supported. Only DWord or String values can be read from the registry
-3	Failed to read the variable value. Please check the security settings

Examples

Tag name	Expression
Tag	GetRegValue(0, "HARDWARE\DESCRIPTION\System", "SystemBios-Date") // Returned value = "08/14/03"
Tag	GetRegValue(2, "Control Panel\Current", "Color Schemes") // Returned value = "Windows Standard "

19.19.22 GetRegValueType

Gets the data type of a registry entry in the Windows registry.

Function	Group	Execution	Windows	Embedded	Thin Client
GetRegValue-Type	System info	synchronous	supported	supported	not supported

Syntax

```
GetRegValueType( numMainKey, strKey, strValueName )
```

Inputs**numMainKey**

Numeric tag to select the main key (see table):

Value	Description
0	HKEY_LOCAL_MACHINE
1	HKEY_CLASSES_ROOT
2	HKEY_CURRENT_USER
3	HKEY_USERS

Appendix: WinStudio-Internal Scripting Language

Value	Description
4	HKEY_CURRENT_CONFIG
5	HKEY_PERFORMANCE_DATA

strKey

Path of the registry entry under the main key

strVariableName

Name of the registry entry the file type of which is to be determined. The maximum length is 255 characters.

Returned value

Value	Description
1	Variable type is String
0	Variable type is DWord
-1	Invalid number of parameters or invalid main key
-2	Variable type is not supported. Only DWord or String values can be read from the registry
-3	Failed to read the variable value. Check whether appropriate security rights are available

Examples

Tag name	Expression
Tag	GetRegValueType(0, "HARDWARE\DESCRIPTION\System", "SystemBiosDate") // Returned value = 1
Tag	GetRegValueType(2, "Control Panel\Desktop", "Smooth Scroll") // Returned value = 0

19.19.23 GetServerHostName

Gets the host name of the server station on which the runtime environment is running.

Function	Group	Execution	Windows	Embedded	Thin Client
GetServerHost-Name	System info	synchronous	not supported	not supported	supported

Syntax

```
GetServerHostName()
```

This function does not have any parameters.

Returned value

Server host name for ISSymbol and 127.0.0.1 for others.

19.19.24 GetTickCount

Gets the current value of the clock counter.

Function	Group	Execution	Windows	Embedded	Thin Client
GetTickCount	System info	synchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Syntax

GetTickCount()

This function does not have any parameters.

Returned value Returns the number of milliseconds counted by the internal clock since initialization of the operating system.

Examples

Tag name	Expression
Tag	GetTickCount() // Returned value = 9400907

19.19.25 InfoAppAlrDir

Returns the file path of the Alarm subfolder of the project.

Function	Group	Execution	Windows	Embedded	Thin Client
InfoAppAlrDir	System info	synchronous	supported	supported	supported

Syntax

InfoAppAlrDir()

This function does not have any parameters.

Returned value Returns the **Alarm** directory of the current project as a string.

Examples

Tag name	Expression
Tag	InfoAppAlrDir() // Returned value = "C:\DemoApp\Alarm\"
Tag	InfoAppAlrDir() // Returned value = "C:\Studio\Projects\App\Alarm\"

19.19.26 InfoAppHSTDir

Returns the file path of the History subfolder of the project.

Function	Group	Execution	Windows	Embedded	Thin Client
InfoAppHSTDir	System info	synchronous	supported	supported	supported

Syntax

InfoAppHstDir()

This function does not have any parameters.

Returned value Returns the **History** directory for the current project as a string.

Examples

Tag name	Expression
Tag	InfoAppHSTDir() // Returned value = "C:\DemoApp\HST\"
Tag	InfoAppHSTDir() // Returned value = "C:\Studio\Projects\App\HST\"

19.19.27 InfoDiskFree

Returns the free disk space on the local computer.

Function	Group	Execution	Windows	Embedded	Thin Client
InfoDiskFree	System info	synchronous	supported	not supported	supported

Syntax

InfoDiskFree(strDisk)

Appendix: WinStudio-Internal Scripting Language

Inputs	strDisk				
	The name of the disk to be checked.				
Returned value	Returns the available free space on the disk in bytes.				
Examples	<table> <tr> <th>Tag name</th><th>Expression</th></tr> <tr> <td>Tag</td><td>InfoDiskFree("C") // Returned value = 2803804605.000000</td></tr> </table>	Tag name	Expression	Tag	InfoDiskFree("C") // Returned value = 2803804605.000000
Tag name	Expression				
Tag	InfoDiskFree("C") // Returned value = 2803804605.000000				

19.19.28 InfoResources

Returns the available resources on the local computer.

Function	Group	Execution	Windows	Embedded	Thin Client
InfoResources	System info	synchronous	supported	supported	supported

Syntax

InfoResources(numSelect)

Inputs

numSelect

Numeric parameter specifying which resource to check:

Value	Description
0	System functions (%)
1	GDI functions (%)
2	USER functions (%)
3	Memory (in bytes)

Examples

Tag name	Expression
Tag	InfoResources(0) // Returned value = 76.000000
Tag	InfoResources(1) // Returned value = 76.000000
Tag	InfoResources(2) // Returned value = 80.000000
Tag	InfoResources(3) // Returned value = 16150528.000000

 The only valid selection on a Windows PC is **3**. Selecting **0**, **1** or **2** only returns **0.000000**.

19.19.29 IsActiveXReg

Checks whether an ActiveX control element is registered on the operating system.

Function	Group	Execution	Windows	Embedded	Thin Client
IsActiveXReg	System info	synchronous	supported	supported	supported

Syntax

IsActiveXReg(numType, strProgIDorFileName)

Appendix: WinStudio-Internal Scripting Language

Inputs

numType

Numeric parameter defining the format of the second **strProgIDorFileName** parameter:

Value	Description
0	Check by program ID
1	Check by file name

strProgIDorFileName

The program ID or file name including path of the ActiveX control element.

Returned value

Value	Description
0	ActiveX is not registered
1	ActiveX is registered

Examples

Tag name	Expression
Tag	IsActiveXReg(0, "ISSYMBOL.ISSymbolCtrl.1") // Returned value = 0
Tag	IsActiveXReg(1, "C:\WinNT\system32\MediaPlayer.ocx") // Returned value = 1

19.19.30 IsAppChangedOnServer

If executed on the client, this function checks whether the project files available on the server are newer than the files currently available on the client.

Function	Group	Execution	Windows	Embedded	Thin Client
IsAppChange- dOnServer	System info	synchronous	supported	supported	supported

Syntax

```
IsAppChangedOnServer({ "optTagUpdateTrigger" })
```

Inputs

optTagUpdateTrigger

An alphanumeric string enclosed in quotation marks or the name of a [String tag, page 113](#) that contains the desired string. This string, in turn, is to be the name of the tag that triggers the function. If the value of this tag changes, the function is automatically executed.

This parameter is optional.



The function is only executed if it is explicitly called, e.g., the [command animation, page 220](#). To execute the function at a regular interval, one of the project [system tags, page 738](#) can be used. For example, **Day** or **Month**.

Returned value

Value	Description
0	No
1	Yes (the files on the server are newer)

Appendix: WinStudio-Internal Scripting Language



For this function, "server" is the station on which the runtime environment is executed and on which the TCP/IP server module is activated. "client" refers to a Thin Client or Secure Viewer communicating with the server via TCP/IP. For further information, refer to [Configuring a web solution, page 391](#).



If the files on the server are newer, i.e., if this function returns TRUE, [ReloadAppFromServer, page 626](#) can be used to update the client.

Examples

Tag name	Expression
Tag	IsAppChangedOnServer()
Tag	IsAppChangedOnServer("CheckVersion") // Function is automatically called when the value of CheckVersion changes.

19.19.31 NoInputTime

Returns the time elapsed since the last keyboard action.

Function	Group	Execution	Windows	Embedded	Thin Client
NoInputTime	System info	synchronous	supported	supported	keyboard input only

Syntax

```
NoInputTime( { "optTagUpdateTrigger" } )
```

Inputs

optTagUpdateTrigger

"Optional" tag that triggers an update if this function is used in a [text output object, page 225](#). Every time this tag changes its value, the returned value is updated.

Returned value

Returns the time (in seconds) since the last keyboard action.

Examples

Tag name	Expression
Tag	NoInputTime()



This function cannot be implemented directly from a text object.

19.19.32 ProductVersion

Returns the number of the WinStudio version.

Function	Group	Execution	Windows	Embedded	Thin Client
ProductVersion	System info	synchronous	supported	supported	supported

Syntax

```
ProductVersion()
```

This function does not have any parameters.

Returned value

Returns the WinStudio version number as a real number.

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	ProductVersion() // Returned value = 7.000000

19.19.33 RegSaveCE

Saves the Windows CE system registry. This function can only be used if the save registry capability is enabled in the Windows CE device image.

Function	Group	Execution	Windows	Embedded	Thin Client
RegSaveCE	System info	synchronous	not supported	supported	not supported

Syntax

RegSaveCE()

This function does not have any parameters.

Returned value

Value	Description
0	Successful
-1	Failed to save HKEY_CLASSES_ROOT
-2	Failed to save HKEY_CURRENT_USER
-3	Failed to save HKEY_LOCAL_MACHINE
-4	Failed to save HKEY_USERS
-5	Function executed on NT platform



This function calls the **RegFlushKey** function from the Windows CE API. The functionality depends on the device firmware of the CE device and is supported by the IndraControl VEx device series (WinCE).

Examples

Tag name	Expression
n/a	RegSaveCE() // Returned value = 0 if successful

19.19.34 ReloadAppFromServer

If executed on the client, this function reloads the necessary project files from the server while maintaining the current state of the project on the client.

Function	Group	Execution	Windows	Embedded	Thin Client
ReloadAppFromServer	System info	synchronous	supported	supported	supported

Syntax

ReloadAppFromServer()

This function does not have any parameters.

Returned value

This function always returns 0.

Appendix: WinStudio-Internal Scripting Language



For this function, "server" means the station on which the runtime environment is actually executed and on which the TCP/IP server module is running. "client" refers to a Thin Client or Secure Viewer that is communicating with the server via TCP/IP. For further information, refer to [Configuring a web solution, page 391](#).



Before calling this function, [IsAppChangedOnServer, page 624](#) can be used to check the version of the project files that are already on the client. If the files on the client are identical to the files on the server, this function does not need to be called.

Examples

Tag name	Expression
Tag	ReloadAppFromServer()

19.19.35 SaveAlarmFile

Use this function to enable/disable the save function for the alarm history and to set the path where the alarm history files are to be saved.

Function	Group	Execution	Windows	Embedded	Thin Client
SaveAlarmFile	System info	synchronous	supported	supported	not supported

Syntax

SaveAlarmFile(numType, optRemotePath)

Inputs

numType

Specifies whether the alarm file is to be saved:

Value	Description
0	Disable saving the alarm file to the local disk
1	Enable saving the alarm file to the local disk
2	Enable saving the alarm file to the local disk and to the remote path specified in the "OptRemotePath" parameter

optRemotePath

Tag containing the path and name of the remote computer where the alarm file is to be saved additionally to the local computer and if **numType** is set to 2.

Returned value

Value	Description
0	Successful
1	2nd parameter is not a String
2	2nd parameter is missing

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	SaveAlarmFile(0)
Tag	SaveAlarmFile(1)
Tag	SaveAlarmFile(2, "Z:\Apps\AppDemo")

19.19.36 SetAppAlarmPath

Sets the "Alarm" path for the current project.

Function	Group	Execution	Windows	Embedded	Thin Client
SetAppAlarmPath	System info	synchronous	supported	supported	executed on the server

Syntax

```
SetAppAlarmPath(strPath)
```

Inputs

strPath

The new **"Alarm"** path for the current project.

Returned value

This function does not return any value.

Examples

Tag name	Expression
	SetAppAlarmPath("C:\Studio\Alarm\ ")

19.19.37 SetAppHSTPath

Sets the file path (directory) where trend history files are to be saved in proprietary format (.HST).

Function	Group	Execution	Windows	Embedded	Thin Client
SetAppHSTPath	System info	synchronous	supported	supported	executed on the server

Syntax

```
SetAppHSTPath(strPath)
```

Inputs

strPath

The file path (directory) where trend history files are to be saved.

Returned value

This function does not return any value.



This function is useful if the file path is to be changed during run-time. The file path can also be set to a network drive by mapping it to the local station or by using the following syntax:

\Network Drive\File Path

Please note that this function does **not** copy existing history files from the default directory to a new one. It only sets the file path for new history files saved after the function is called.

Examples

Tag name	Expression
	SetAppHstPath("C:\Studio\History\")

19.19.38 SetDateFormat

Sets the format and separator for the date String.

Function	Group	Execution	Windows	Embedded	Thin Client
SetDateFormat	System info	synchronous	supported	supported	supported

Syntax

```
SetDateFormat( strSeparator, strFormat )
```

Inputs

strSeparator

The separator for the date String.

strFormat

String tag specifying the order of Month (M), Day (D) and Year (Y) in the date String.

Value	Description
DMY	Day, Month, Year
MDY	Month, Day, Year
YMD	Year, Month, Day

Returned value

Value	Description
0	No error
1	Invalid parameter

Examples

Tag name	Expression
Tag	SetDateFormat("/", "MDY") // Date = 04/18/2002
Tag	SetDateFormat(":", "MYD") // Date = 04:2002:18

19.19.39 SetKeyboardLanguage

Sets the language of the virtual keyboards in the project.

Function	Group	Execution	Windows	Embedded	Thin Client
SetKeyboardLanguage	System info	synchronous	supported	supported	supported

Syntax

```
SetKeyboardLanguage(strLanguage)
```

Inputs

strLanguage

String tag with the language code used for the virtual keyboards. The currently available options include:

Appendix: WinStudio-Internal Scripting Language

	Value	Description
	EN	English (default)
	GE	German
Returned value	Value	Description
	0	Successful
	1	Error
Examples	Tag name	Expression
	Tag	SetKeyboardLanguage("EN")
	Tag	SetKeyboardLanguage(tagLanguage)

19.19.40 SetRegValue

Sets the value of a registry entry.

Function	Group	Execution	Windows	Embedded	Thin Client
SetRegValue	System info	synchronous	supported	supported	not supported

Syntax

SetRegValue(numMainKey, strKey, strVariableName, numType, strOrNumValue)

Inputs**numMainKey**

Numeric tag to select the main key (see table):

Value	Description
0	HKEY_LOCAL_MACHINE
1	HKEY_CLASSES_ROOT
2	HKEY_CURRENT_USER
3	HKEY_USERS
4	HKEY_CURRENT_CONFIG
5	HKEY_PERFORMANCE_DATA

strKey

Path of the registry entry under the main key

strVariableName

Name of the registry entry to be changed. The maximum length is 255 characters.

numType

Format of the registry entry to be changed (currently supported: see table)

Appendix: WinStudio-Internal Scripting Language

Value	Description
0	DWord
1	Character string

strOrNumValue

Value to be written (String variable or string in quotation marks).

Returned value

Value	Description
0	Successful
-1	Invalid number of parameters or invalid main key
-2	Invalid type
-3	Failed to read the registry key. Check whether appropriate security rights are available

Examples

Tag name	Expression
Tag	SetRegValue(0, "HARDWARE\DEVICEMAP\SERIALCOMM", "\Device\nSerial1", 1, "COM3") // Returned value = 0 if successful
Tag	SetRegValue(2, "Control Panel\Desktop", "Smooth Scroll", 0, 1) // Returned value = 0 if successful

NOTICE

This entry can affect the Windows system configuration.

Proceed with extreme caution and "only" edit the registry in case of no doubt about the configuration.

19.19.41 SetWebConfig

Configures the web settings for the current project. This function updates the settings in the HTML files of the project.

Function	Group	Execution	Windows	Embedded	Thin Client
SetWebConfig	System info	synchronous	supported	supported	executed on the server

Syntax

```
SetWebConfig( strServerIP, optStrBackupURL, optStrPathFile, optNumHostPort, optStrSecondaryServerIP,
              optNumProtocolFlag, optNumGtwPort, optStrGtwIP, optStrSecondaryGtwIP, optStrISSymbolURL )
```

Inputs

strServerIP

Data server IP address. IP address (or hostname) of the computer where the TCP Server module of "WinStudio" is running.

optStrBackupURL

Alternative URL for the web pages of the project (optional). The Thin Client searches for the web pages under this URL if they cannot be found under the same URL written in the address field of the web browser.

Appendix: WinStudio-Internal Scripting Language

optStrPathFile

File path and name of the HTML file to be updated (optional). If only the file path without a file name is specified, all of the HTML files at the specified file path are updated.



An individual file can be specified (e.g., if only one file is to be updated.) This is particularly useful for projects running on Windows CE target systems.

optNumHostPort

TCP port number to be used by the Thin Client to exchange data with the TCP server module of "WinStudio" (optional).

optStrSecondaryServerIP

Alternative data server IP address (optional). The Thin Client attempts to connect to the TCP server module of "WinStudio" at this IP address if it cannot connect to the TCP Server module running at the IP address specified in the strServerIP parameter.

optNumProtocolFlag

If the Web Tunneling Gateway option is used, this parameter specifies whether the Thin Client uses HTTP or HTTPS (SSL – Secure Socket Layer) to exchange data with the web server (optional). If this flag has the value 0, the Thin Client uses HTTP. If this flag has the value 1, the Thin Client uses HTTPS (SSL).

optNumGtwPort

TCP port number to be used by the Thin Client to exchange data with the web server when using the Web Tunneling Gateway.

optStrGtwIP

IP address (or hostname) of the computer where the Web Tunneling Gateway is running (optional).

optStrSecondaryGtwIP

Alternative IP address of the computer where the Web Tunneling Gateway is running (optional). The Thin Client attempts to connect to the Web Tunneling Gateway at this IP address if it cannot connect to the Web Tunneling Gateway running at the IP address specified in the optStrGtwIP parameter.

optStrISSymbolURL

URL from where the updated version of ISSymbol (ActiveX control element) has to be downloaded if it is not properly registered on the Thin Client station (optional).

Appendix: WinStudio-Internal Scripting Language



- Tags or expressions can be used as arguments in this function. For this reason, this function can be used to configure the web settings automatically during runtime, depending on the network settings of each project (IP address, web server URL, etc.).
- Only the first parameter of this function ("strServerIP") has to be specified. All other parameters are optional. The parameters that are not configured in the function assume the default value configured on the [Web, page 78](#) tab of the project settings.
- The following parameters are only to be used if the Web Tunneling Gateway is used: "optNumProtocolFlag", "optNumGtwPort", "optStrGtwIP", "optStrSecondaryGtwIP" and "optStrISSymbolURL".

Returned value

Error	Description
0	No error
1	Invalid number of parameters
2	Invalid Server IP address 1
3	Invalid URL
4	Invalid optional path
5	No web pages found

Examples

Tag name	Expression
Tag	SetWebConfig("192.168.1.28")
Tag	SetWebConfig("192.168.1.28", "http://192.168.1.28/")
Tag	SetWebConfig(GetComputerIP(), "http://" + GetComputerIP() + "/")
Tag	SetWebConfig("192.168.1.28", "http://192.168.1.28/", "C:\MyWebPages\")
Tag	SetWebConfig("192.168.1.28", "http://192.168.1.28/", "C:\MyWebPages\", 1234)
Tag	SetWebConfig("192.168.1.28", "http://200.0.0.10/", "C:\MyWebPages\", 1234, "192.168.1.29", 0, 80, "200.0.0.1", "200.0.0.10", "http://200.0.0.10/MyISSymbol/")

19.19.42 SNMPGet

Gets information from computers or network devices via the Simple Network Management Protocol (SNMP).

Function	Group	Execution	Windows	Embedded	Thin Client
SNMPGet	System info	synchronous	supported	not supported	supported

Syntax

```
SNMPGet( strAddress, strCommunity, strOID, "strTagName" )
```

Inputs

strAddress

The address of the device/computer (e.g., "127.0.0.1" or "localhost").

Appendix: WinStudio-Internal Scripting Language

strCommunity

SNMP community name when communicating with the computer (e.g., "public").

strOID

OID to be consulted (e.g., "1.3.6.1.2.1.1.1.0").

strTagName

Name of the tag that is to receive the requested value.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

Value	Description
0	No error
-1	Invalid number of parameters
-2	Invalid parameter
-3	Cannot connect to the remote device
-4	Cannot connect to the remote device
-5	GET operation failed
-6	Invalid OID
-7	Invalid tag name
-8	Invalid tag type
-9	This function is not supported by the current OS

Examples

Tag name	Expression
ErrorTag	SNMPGet("127.0.0.1", "public", "1.3.6.1.2.1.1.1.0", "SysDescrTag") // ErrorTag will receive the error code. If the function succeeds, the value in the OID "1.3.6.1.2.1.1.1.0" will be saved in the tag SysDescrTag.

19.19.43 SNMPSet

Uses the Simple Network Management Protocol (SNMP) to set a value on a target computer or network device.

Function	Group	Execution	Windows	Embedded	Thin Client
SNMPSet	System info	synchronous	supported	not supported	supported

Syntax

```
SNMPSet( strAddress, strCommunity, strOID, Value, optNumType )
```

Inputs**strAddress**

The address of the target computer or device (e.g., "127.0.0.1" or "localhost").

Appendix: WinStudio-Internal Scripting Language

strCommunity

The SNMP community name (e.g., "public") when communicating with the target computer or device.

strOID

The object ID (OID) to be set.

Value

The value to be set to the specified OID.

optNumType

A numeric value or tag of type [Integer, page 113](#) specifying the data type. This is an optional parameter, however, if it is included, it requires one of following values:

Value	Type	Description
0	OCTETSTRING	An Octet String variable
1	INTEGER32	A 32-bit signed Integer variable
2	TIMETICKS	A time ticks variable
3	GAUGE32	A gauge variable
4	COUNTER32	A counter variable
5	IPADDRESS	An IP address variable
6	OBJECTIDENTIFIER	An object identifier variable
7	SEQUENCE	An ASN sequence variable
8	OPAQUE	A non-transparent variable

Returned value

Value	Description
0	No error
-1	Invalid number of parameters
-2	Invalid parameter
-3	Cannot connect to the remote device
-4	Cannot connect to the remote device
-5	SET operation failed
-6	Invalid OID
-7	Invalid tag name
-8	Invalid tag type
-9	This function is not supported by the current OS

Examples

Tag name	Expression
	SNMPSet("127.0.0.1", "public", ".1.3.6.1.2.1.1.0", 123, 1) //Sets an integer value of 123 to the specified OID on the localhost (127.0.0.1).

Appendix: WinStudio-Internal Scripting Language

19.19.44 WritePrivateProfileString

Writes a specified setting to the project viewer initialization file, using the standard format of the .ini file.

Function	Group	Execution	Windows	Embedded	Thin Client
WritePrivateProfileString	System info	synchronous	supported	supported	supported

Syntax

```
WritePrivateProfileString ( strSection, strName, strValue, strFileName )
```

Inputs**strSection**

The section name of the ini file to be written.

strName

Parameter within the selected section to be written.

strValue

Value to be written.

strFileName

Name and path of the .ini file to be written.

Returned value The function returns 1 if the file was updated successfully.



When running on Windows CE devices, this function rewrites the entire file. For this reason, it is not recommended to use this function for long files on Windows CE devices. If it is executed on a Windows CE device, the function also adds the following lines to the end of the file: [FileBackUpControl] Valid=1

Examples

Tag name	Expression
Tag	WritePrivateProfileString(Section, Name, Value, FileName)
Tag	WritePrivateProfileString("Options", "ds1", "Value", "C:\Viewer.ini")

19.20 Tags Database Functions

19.20.1 Tags Database Functions – General Information

These functions are used to directly change the values of project tags.

Tags database

Function	Execution	Windows PC	Windows CE	Thin Client
ExecuteAlarmAck, page 637	synchronous	supported	supported	supported
ForceTagChange, page 638	synchronous	supported	supported	supported
GetTagValue, page 638	synchronous	supported	supported	supported
SetTagValue, page 639	synchronous	supported	supported	supported

19.20.2 ExecuteAlarmAck

This function acknowledges an active alarm with the specified tag. The advantage of using this function is that if used from the Thin Client, the Alarm task stores the user name and station from which the alarm was acknowledged.

Function	Group	Execution	Windows	Embedded	Thin Client
ExecuteAlarmAck	Tags database	synchronous	supported	supported	supported

Syntax

```
ExecuteAlarmAck( "strTagName", optStrComment, optStrAlarmType )
ExecuteAlarmAck("strTagName",{ | ,optStrComment{ | ,optStrAlarmType } })
```

Inputs

strTagName

Name of the tag the alarm of which is to be acknowledged.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

optStrComment

An optional comment to send to the Alarm task along with the user name and station.

optStrAlarmType

If more than one alarm is active on the specified tag, the alarm (e.g., Hi, Lo, HiHi, LoLo) to be acknowledged can be specified. Otherwise, the function acknowledges the most recently activated alarm.

Returned value

Value	Description
0	Successfully executed
-1	Invalid number of parameters
-2	Invalid tag name
-3	Executed, but did not wait for confirmation from Alarms task. See note



If this function is used to acknowledge an alarm, it typically waits for an acknowledgement by the Alarm task before returning a value of "0" to indicate successful execution. In some cases, however, waiting for acknowledgement may cause the runtime environment to crash. If this is the case and if the function is properly called with valid parameters, it executes as intended, however, does not wait for acknowledgement.

Examples

Acknowledging the active Hi alarm on tag **A** with the comment **Hi alarm on tag A**:

```
ExecuteAlarmAck ( "A", "Hi alarm on tag A", "Hi" )
```

Appendix: WinStudio-Internal Scripting Language

19.20.3 ForceTagChange

Forces writing of a value to a tag in the database so that the new value is recognized as a change in value if it has not changed compared to the old value.

Function	Group	Execution	Windows	Embedded	Thin Client
ForceTagChange	Tags database	synchronous	supported	supported	supported

Syntax

```
ForceTagChange( "strTagName", numValue )
```

Inputs**strTagName**

The name of the tag forced to accept the new value.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

numValue

The new value to be written to the specified tag.

Returned value

This function does not return any value.

Examples

Tag name	Expression
n/a	ForceTagChange("TagA", 5)

19.20.4 GetTagValue

Gets the value of the specified tag from the project tags database.

Function	Group	Execution	Windows	Embedded	Thin Client
GetTagValue	Tags database	synchronous	supported	supported	supported

Syntax

```
GetTagValue( "strTagName", optNumRefresh )
```

Inputs**strTagName**

The name of the tag the value of which is to be retrieved.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

optNumRefresh

The tag to be used as a trigger to update the function. If the value of the specified tag changes, the function is executed again.

Appendix: WinStudio-Internal Scripting Language



Normally, a function is only executed if the object on which it is configured changes in some way, e.g., if a [pushbutton object, page 178](#) is clicked. To execute the function at a regular interval, one of the [system tags, page 110](#) can be used. For example, "Second", "Minute" or "Hour".

Returned value

This function only returns the value of the tag specified by **strTagName**. If the specified tag does not exist, the function returns the value 0.

The value of the tag specified by **optNumRefresh** does not affect the value returned by the function in any way.

Examples

Tag name	Expression
requiredTag = 15	GetTagName("requiredTag", Second) // Return = 15
pointerTag = "Required Tag" requiredTag = 15 optNum = Second	GetTagName(pointerTag, optNum) // Return = 15

19.20.5 SetTagValue

Sets the value of the specified tag in the project tags database.

Function	Group	Execution	Windows	Embedded	Thin Client
SetTagValue	Tags database	synchronous	supported	supported	supported

Syntax

```
SetTagValue( "strTagName", TagValue )
```

Inputs

strTagName

The name of the tag to be set.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

TagValue

The new value to be set to the specified tag.

Returned value

Value	Description
-1	Invalid tag name
0	No error

Examples

Tag name	Expression
TagA	SetTagValue("TagA", "Hello") // Return = Hello
TagA	SetTagValue("TagA", 123) // Return = 123
TagA, TagB= 15	SetTagValue("TagA", TagB) // Return = 15

Appendix: WinStudio-Internal Scripting Language

19.21 Loop Functions

19.21.1 Loop Functions – General Information

These functions are used to implement repeating, incrementing loops within a script.

Loop

Function	Execution	Windows PC	Windows CE	Thin Client
For , page 640 & "Next"	N/A	supported	supported	not supported

19.21.2 For ... Next

Implements a **For ... Next** loop within a math worksheet or dynamic object. The section of the script affected by the loop begins with the **For()** function call and ends with the **Next** element. The **Next** element directs back to the beginning of the loop.

Function	Group	Execution	Windows	Embedded	Thin Client
For ... Next	Loop	N/A	supported	supported	supported

Syntax

```
For( numInitialValue, numFinalValue, numStep ) ... Next
```

Inputs

numInitialValue

The initial step (increment) of the loop.

numFinalValue

The final step (the final increment) of the loop.

numStep

The step (increment) of the loop.

Returned value

Returns the step on which the loop is currently running.

Examples

Tag name	Expression
Tag	For(1, 5, 1)
Next	



Every **For** function has to be combined with a **Next** element. As shown in the example, the **Next** element has to be specified in the tag field of the math worksheet.

19.22 ODBC Functions

19.22.1 ODBC Functions – General Information

These functions are used to interact with an external database via Open Database Connectivity (ODBC).

Appendix: WinStudio-Internal Scripting Language



These functions are provided to support legacy database interfaces. In most cases, it is recommended to use the newer [data-base/ERP connections manager, page 328](#) and the respective functions, [page 692](#).

ODBC

Function	Execution	Windows PC	Windows CE	Thin Client
ODBCBeginTrans, page 642	synchronous	supported	not supported	executed on the server
ODBCBindCol, page 642	synchronous	supported	not supported	executed on the server
ODBCCanAppend, page 644	synchronous	supported	not supported	executed on the server
ODBCCanTransact, page 644	synchronous	supported	not supported	executed on the server
ODBCCanUpdate, page 644	synchronous	supported	not supported	executed on the server
ODBCClose, page 645	synchronous	supported	not supported	executed on the server
ODBCCommitTrans, page 645	synchronous	supported	not supported	executed on the server
ODBCDelete, page 646	synchronous	supported	not supported	executed on the server
ODBCExecuteSQL, page 646	synchronous	supported	not supported	executed on the server
ODBCInsert, page 647	synchronous	supported	not supported	executed on the server
ODBCIsBOF, page 648	synchronous	supported	not supported	executed on the server
ODBCIsDeleted, page 648	synchronous	supported	not supported	executed on the server
ODBCIsEOF, page 649	synchronous	supported	not supported	executed on the server
ODBCIsFieldNULL, page 649	synchronous	supported	not supported	executed on the server
ODBCIsFieldNullable, page 650	synchronous	supported	not supported	executed on the server
ODBCMove, page 650	synchronous	supported	not supported	executed on the server
ODBCMoveFirst, page 651	synchronous	supported	not supported	executed on the server
ODBCMoveLast, page 651	synchronous	supported	not supported	executed on the server
ODBCMoveNext, page 652	synchronous	supported	not supported	executed on the server

Appendix: WinStudio-Internal Scripting Language

Function	Execution	Windows PC	Windows CE	Thin Client
ODBCMovePrev, page 652	synchronous	supported	not supported	executed on the server
ODBCOpen, page 653	synchronous	supported	not supported	executed on the server
ODBCQuery, page 654	synchronous	supported	not supported	executed on the server
ODBCRollback, page 654	synchronous	supported	not supported	executed on the server
ODBCSetFieldNULL, page 655	synchronous	supported	not supported	executed on the server
ODBCSetFilter, page 656	synchronous	supported	not supported	executed on the server
ODBCSetSort, page 656	synchronous	supported	not supported	executed on the server
ODBCUnbindCol, page 657	synchronous	supported	not supported	executed on the server
ODBCUpdate, page 657	synchronous	supported	not supported	executed on the server

19.22.2 ODBCBeginTrans

Begins a transaction with the connected data source.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCBeginTrans	ODBC	synchronous	supported	not supported	executed on the server

Syntax

```
ODBCBeginTrans(numHandler)
```

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Database not open
3	Error beginning transaction

Examples

Tag name	Expression
Tag	ODBCBeginTrans(5)

19.22.3 ODBCBindCol

Binds a column to a tag.

Appendix: WinStudio-Internal Scripting Language

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCBindCol	ODBC	synchronous	supported	not supported	executed on the server

Syntax

```
ODBCBindCol( numHandler, strColName, strColType, strTagName )
```

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

strColName

The database column name.

strColType

The SQL data type (of the following list):

- SQL_BIT
- SQL_TINYINT
- SQL_LONGVARCHAR
- SQL_CHAR
- SQL_VARCHAR
- SQL_DECIMAL
- SQL_NUMERIC
- SQL_DATE
- SQL_TIME
- SQL_TIMESTAMP
- SQL_DOUBLE
- SQL_REAL
- SQL_SMALLINT
- SQL_INTEGER

strTagName

The name of the tag to be bound to the column.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Invalid parameter type
3	One of the parameters has an empty string
4	ColType contains an invalid type



Every time after a column is bound, the [ODBCQuery, page 654](#) function has to be called.

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	ODBCBindCol(5, "OrderDate", "SQL_DATE", "Order_Date")

For further information, refer to: [ODBCUnbindCol\(\)](#), [page 657](#)

19.22.4 ODBCCanAppend

Checks whether the database enables adding new records.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCCanAppend	ODBC	synchronous	supported	not supported	executed on the server

Syntax

```
ODBCCanAppend( numHandler )
```

Inputs

numHandler

The handler returned by the [ODBCOpen](#), [page 653](#) function.

Returned value

Value	Description
0	Database does not enable adding new records
Not zero	Database does enable adding new records

Examples

Tag name	Expression
Tag	ODBCCanAppend(5)

19.22.5 ODBCCanTransact

Checks whether the database enables transactions.

Function	Group	Execution	Windows	Embedded	Thin Client
Avg	ODBC	synchronous	supported	not supported	executed on the server

Syntax

```
ODBCCanTransact( numHandler )
```

Inputs

numHandler

The handler returned by the [ODBCOpen](#), [page 653](#) function.

Returned value

Value	Description
0	Database does not enable transactions
Not zero	Database does enable transactions

Examples

Tag name	Expression
Tag	ODBCCanTransact(2)

19.22.6 ODBCCanUpdate

Checks whether the database can be updated.

Appendix: WinStudio-Internal Scripting Language

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCCanUpdate	ODBC	synchronous	supported	not supported	executed on the server

Syntax

ODBCCanUpdate(numHandler)

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

Returned value

Value	Description
0	Database does not enable updates
Not zero	Database does enable updates

Examples

Tag name	Expression
Tag	ODBCCanUpdate(6)

19.22.7 ODBCClose

Closes a connection to the database.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCClose	ODBC	synchronous	supported	not supported	executed on the server

Syntax

ODBCClose(numHandler)

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

Returned value

Value	Description
0	Successful
1	Invalid handler

Examples

Tag name	Expression
Tag	ODBCClose(5)

19.22.8 ODBCCommitTrans

Commits a transaction. Call this function after completing transactions.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCCommit-Trans	ODBC	synchronous	supported	not supported	executed on the server

Syntax

ODBCCommitTrans(numHandler)

Appendix: WinStudio-Internal Scripting Language

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Database not open
3	Error committing transaction

Examples

Tag name	Expression
Tag	ODBCCommitTrans(1)

19.22.9 ODBCDelete

Deletes the current record.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCDelete	ODBC	synchronous	supported	not supported	executed on the server

Syntax

```
ODBCDelete(numHandler)
```

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Database not open
3	Delete error



After successful deletion, one of the "move" functions has to be called explicitly (i.e., [ODBCMove, page 650](#), [ODBCMoveFirst, page 651](#), [ODBCMoveLast, page 651](#), [ODBCMoveNext, page 652](#), [ODBCMovePrev, page 652](#)) to exit the deleted record.

Examples

Tag name	Expression
Tag	ODBCDelete(5)

19.22.10 ODBCExecuteSQL

Directly executes an SQL statement.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCExecuteSQL	ODBC	synchronous	supported	not supported	executed on the server

Appendix: WinStudio-Internal Scripting Language

Syntax

```
ODBCExecuteSQL( numHandler, strSqlCommand )
```

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

strSqlCommand

A valid SQL statement.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Database not open
3	Invalid parameter
4	Error executing SQL command



This function does not return any records, regardless of the statement that is executed.

Examples

Tag name	Expression
Tag	ODBCExecuteSQL(3, ")
Tag	ODBCExecuteSQL(4, ")

19.22.11 ODBCInsert

Inserts a new record into the database.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCInsert	ODBC	synchronous	supported	not supported	executed on the server

Syntax

```
ODBCInsert(numHandler)
```

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Database not open
3	Insert error



This function uses the values of the tags linked by the [ODBC-BindCol, page 642](#) function to create the new record.

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	ODBCInsert(7)

19.22.12 ODBCIsBOF

Returns whether the first record of the record set was exceeded. (Call this function before browsing through the records.)

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCIsBOF	ODBC	synchronous	supported	not supported	executed on the server

Syntax

```
ODBCIsBOF(numHandler)
```

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

Returned value

Value	Description
0	Record found
Not zero	Record set contains no records or it is moved backward, exceeding the first record

This function can also be used in combination with the [ODBCIsEOF, page 649](#) function to determine whether the record set contains any records or is empty. Immediately after calling the [ODBCQuery, page 654](#) function and if the record set contains no records, **ODBCIsBOF** returns a non-zero value. If a record set with at least one record is opened, the first record is the current record and **ODBCIsBOF** returns zero ("0"). If the first record is the current record, and [ODBCMovePrev, page 652](#) is called, **ODBCIsBOF** returns a non-zero value.

Examples

Tag name	Expression
Tag	ODBCIsBOF(1)

19.22.13 ODBCIsDeleted

Reports whether the current record was deleted.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCIsDeleted	ODBC	synchronous	supported	not supported	executed on the server

Syntax

```
ODBCIsDeleted(numHandler)
```

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

Returned value

Value	Description
0	Record set is not positioned on a deleted record.
Not zero	Record set is positioned on a deleted record

Appendix: WinStudio-Internal Scripting Language

If it is moved to a record and this function returns a non-zero value, it has to be switched to another record before any other operations can be performed.

Examples

Tag name	Expression
Tag	ODBCIsDeleted(8)

19.22.14 ODBCIsEOF

Reports whether the last record of the record set was exceeded. (Call this function on browsing through the records.)

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCIsEOF	ODBC	synchronous	supported	not supported	executed on the server

Syntax

ODBCIsEOF(numHandler)

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

Returned value

Value	Description
0	Record found
Not zero	Record set contains no records or it is moved forward, exceeding the last record.

This function can be used in combination with the [ODBCIsBOF, page 648](#) function to determine whether the record set contains any records or is empty. Immediately after calling the [ODBCQuery, page 654](#) function and if the record set contains no records, **ODBCIsBOF** returns a non-zero value. If a record set with at least one record is opened, the first record is the current record and **ODBCIsEOF** returns a zero ("0"). If the last record is the current record and [ODBCMoveNext, page 652](#) is called, **ODBCIsEOF** returns a non-zero value.

Examples

Tag name	Expression
Tag	ODBCIsEOF(5)

19.22.15 ODBCIsFieldNULL

Reports whether a specified field in a record set was flagged as "ZERO".

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCIsField-NULL	ODBC	synchronous	supported	not supported	executed on the server

Syntax

ODBCIsFieldNULL(numHandler, strColName)

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

Appendix: WinStudio-Internal Scripting Language

strColName

The column name.

Returned value

Value	Description
0	The specified field is not flagged as Null
Not zero	The specified field is flagged as Null

Examples

Tag name	Expression
Tag	ODBCIsFieldNULL(7, "CustomerName")
Tag	ODBCIsFieldNULL(3, "CompanyName")

19.22.16 ODBCIsFieldNullable

Checks whether a specified field is nullable (i.e., can be set to a "NULL" value).

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCIsFieldNullable	ODBC	synchronous	supported	not supported	executed on the server

Syntax

```
ODBCIsFieldNullable(numHandler, strColName)
```

Inputs**numHandler**The handler returned by the [ODBCOpen, page 653](#) function.**strColName**

The column name.

Returned value

Value	Description
0	The specified field is not flagged as Nullable
Not zero	The specified field is flagged as Nullable

Examples

Tag name	Expression
Tag	ODBCIsFieldNullable(1, "Price")
Tag	ODBCIsFieldNullable(1, "Model")

19.22.17 ODBCMove

Moves the current record pointer either forward or backward within a record set.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCMove	ODBC	synchronous	supported	not supported	executed on the server

Syntax

```
ODBCMove(numHandler, numOffset)
```

Appendix: WinStudio-Internal Scripting Language

Inputs

numHandler
The handler returned by the ODBCOpen, page 653 function.
numOffset
The number of rows to move forward or backward:
- Positive values move forward towards the end of the record set. - Negative values move backward towards the beginning of the record set. - A value of "0" updates the current record.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Database not open
3	Move error

Examples

Tag name	Expression
Tag	ODBCMove(2, 3)
Tag	ODBCMove(8, 2)

19.22.18 ODBCMoveFirst

Moves to the first record within the record set.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCMoveFirst	ODBC	synchronous	supported	not supported	executed on the server

Syntax

ODBCMoveFirst(numHandler)

Inputs

numHandler
The handler returned by the ODBCOpen, page 653 function.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Database not open
3	Move error

Examples

Tag name	Expression
Tag	ODBCMoveFirst(4)

19.22.19 ODBCMoveLast

Moves the pointer to the last record within the record set.

Appendix: WinStudio-Internal Scripting Language

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCMoveLast	ODBC	synchronous	supported	not supported	executed on the server

Syntax

```
ODBCMoveLast ( numHandler )
```

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Database not open
3	Move error

Examples

Tag name	Expression
Tag	ODBCMoveLast(7)

19.22.20 ODBCMoveNext

Moves the pointer to the next record within the record set.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCMoveNext	ODBC	synchronous	supported	not supported	executed on the server

Syntax

```
ODBCMoveNext ( numHandler )
```

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Database not open
3	End of record set reached
4	Move error

Examples

Tag name	Expression
Tag	ODBCMoveNext(9)

19.22.21 ODBCMovePrev

Moves the pointer to the next record within the record set.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCMovePrev	ODBC	synchronous	supported	not supported	executed on the server

Appendix: WinStudio-Internal Scripting Language

Syntax

ODBCMovePrev(numHandler)

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Database not open
3	Beginning of record set reached
4	Move error

Examples

Tag name	Expression
Tag	ODBCMovePrev(2)

19.22.22 ODBCOpen

Opens a connection to the database and returns a numeric handler to be used by other ODBC functions.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCOpen	ODBC	synchronous	supported	not supported	executed on the server

Syntax

ODBCOpen(strDsn, strUser, strPassw, strTable, strFilter, strSort)

Inputs

strDsn

The name of the data source.

strUser

The user name.

strPassw

The password.

strTable

The name of the database table.

strFilter

The SQL **WHERE** clause.

strSort

The SQL **ORDER BY** clause.

Appendix: WinStudio-Internal Scripting Language

Returned value

Value	Description
n	On successful execution, returns the handler to identify the database
-1	Invalid parameter
-2	DSN or TableName contain an empty string



This function does not read or write any data. It simply creates a handler to manipulate the database. The columns have to be bound by using the [ODBCBindCol, page 642](#) function and the [ODBCQuery, page 654](#) function has to be called to retrieve the first record.

Examples

Tag name	Expression
Tag	ODBCOpen("MyDSNFile", "Alex", "", "Table1", "Name='Mayer'", "Name ASC")
Tag	ODBCOpen("DSNFileName", "Robert", "Robot", "Table1", "", "")

19.22.23 ODBCQuery

Retrieves the currently selected record from a database.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCQuery	ODBC	synchronous	supported	not supported	executed on the server

Syntax

```
ODBCQuery(numHandler)
```

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	No columns bound
3	Cannot open database
4	Cannot restart database
5	Query error



If [column binding, page 642](#) or [filter, page 656](#) and [sorting, page 656](#) is changed, this function has to be called again.

Examples

Tag name	Expression
Tag	ODBCQuery(6)

19.22.24 ODBCRollBack

Discards the changes made during a transaction.

Appendix: WinStudio-Internal Scripting Language

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCRollBack	ODBC	synchronous	supported	not supported	executed on the server

Syntax

ODBCRollBack(numHandler)

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Database not open
3	Error discarding the changes

Examples

Tag name	Expression
Tag	ODBCRollback(4)

19.22.25 ODBCSetFieldNULL

Flags a field data member in the record set as "NULL" (specifically without value) or as "not NULL".

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCSetField-NULL	ODBC	synchronous	supported	not supported	executed on the server

Syntax

ODBCSetFieldNULL(numHandler, strColName, numValue)

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

strColName

The column name.

numValue

A numeric tag that specifies the field data as NULL if 0 and not NULL if not zero.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Database not open
3	Invalid parameter
4	Invalid column name

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	ODBCSetFieldNULL(2, "Price", 1)
Tag	ODBCSetFieldNULL(4, "CompanyName", 0)

19.22.26 ODBCSetFilter

Limits the records selected in a database.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCSetFilter	ODBC	synchronous	supported	not supported	executed on the server

Syntax

ODBCSetSort(numHandler, strSort)

Inputs

numHandlerThe handler returned by the [ODBCOpen, page 653](#) function.**strFilter**The SQL **WHERE** clause.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Invalid parameter



This function can be used for selecting a subordinate set of records, e.g., "all salespersons based in California" ("**state** = 'CA'"). After calling this function, [ODBCQuery, page 654](#) has to be called.

Examples

Tag name	Expression
Tag	ODBCSetFilter(3, "Name='Morgan'")

19.22.27 ODBCSetSort

Sorts the records selected in a database.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCSetSort	ODBC	synchronous	supported	not supported	executed on the server

Syntax

ODBCSetSort(numHandler, strSort)

Inputs

numHandlerThe handler returned by the [ODBCOpen, page 653](#) function.

Appendix: WinStudio-Internal Scripting Language

strSort

The SQL **ORDER BY** clause.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Invalid parameter type



This property can be used to sort records in one or more columns. After calling this function, [ODBCQuery, page 654](#) has to be called.

Examples

Tag name	Expression
Tag	ODBCSetSort(5, "Name DESC")

19.22.28 ODBCUnbindCol

Unbinds a column that was previously linked by the "ODBCBindCol" function.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCUnbindCol	ODBC	synchronous	supported	not supported	executed on the server

Syntax

ODBCUnbindCol(numHandler, strColName)

Inputs

numHandler

The handler returned by the [ODBCOpen, page 653](#) function.

strColName

The column name.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Invalid parameter type
3	Column not bound

Examples

Tag name	Expression
Tag	ODBCUnbindCol(7, "Name")

19.22.29 ODBCUpdate

Updates the current record.

Function	Group	Execution	Windows	Embedded	Thin Client
ODBCUpdate	ODBC	synchronous	supported	not supported	executed on the server

Appendix: WinStudio-Internal Scripting Language

Syntax

 ODBCUpdate(numHandler)

Inputs

numHandler

 The handler returned by the [ODBCOpen, page 653](#) function.

Returned value

Value	Description
0	Successful
1	Invalid handler
2	Database not open
3	Update error


 This function uses the values of the tags linked by the [ODBC-BindCol, page 642](#) function to update the current record.

Examples

Tag name	Expression
Tag	ODBCUpdate(1)

19.23 Email Functions

19.23.1 Email Functions – General Information

These functions are used to configure and send emails from within a project.

Email

Function	Execution	Windows PC	Windows CE	Thin Client
CnfEmail, page 658	synchronous	supported	supported	supported
GetStatusSendEmailExt, page 661	synchronous	supported	supported	supported
SendEmail, page 661	synchronous	supported	supported	supported
SendEmailExt, page 662	asynchronous	supported	supported	supported

19.23.2 CnfEmail

 This function configures the email settings used by other features in the project that can send emails. For example, alarm worksheets and the **SendEmail** and **SendEmailExt** functions.

Function	Group	Execution	Windows	Embedded	Thin Client
CnfEmail	Email	synchronous	supported	supported, except for encryption	supported

Syntax

 CnfEmail(strSMTP, strFrom, strPOP3, strUser, strPassword, optNumTimeout, optNumAuthType, optStrSMTPUser, optStrSMTPPassword)

 CnfEmail(strSMTP, strFrom, strPOP3, strUser, strPassword{ | , optNumTimeout{ | , { optNumAuthType | 1 | 2 } { | , optStrSMTPUser, optStrSMTPPassword } } })

Appendix: WinStudio-Internal Scripting Language

Inputs

strSMTP

The hostname or IP address of the outgoing email server, which is also referred to as SMTP server. A port number can be specified if the server does not use any of the standard SMTP ports.



On Windows CE devices, an IP address has to be specified.

strFrom

The email address to send and receive emails. This is to be a valid address on the POP3 server (see **strPOP3** in the following).

strPOP3

The hostname or IP address of the incoming email server, also referred to as the POP3 server. A port number can be specified if the server does not use any of the standard POP3 ports.



On Windows CE devices, an IP address has to be specified.

strUser

The user name to be used to log onto the POP3 server.

strPassword

The password to be used to log onto the POP3 server.

optNumTimeout

The timeout limit (in seconds) for sending emails. If no response is received from the SMTP server within this period of time, the operation is aborted.

This is an optional parameter. If no timeout is specified, the project keeps trying to send until a response is received. However, a timeout limit is to be specified to ensure that the project does not freeze.

optNumAuthType

optStrSMTPUser

optStrSMTPPassword

By default, SMTP servers do not require authentication for outgoing emails. If the server does require authentication, set **optNumAuthType** to "1" (unencrypted) or "2" (encrypted via TLS/SSL) and specify the user name and password. (If the SMTP and POP3 user name and password are identical, **optStrSMTPUser** and **optStrSMTPPassword** can be skipped. In this case, the project automatically uses the values from **strUser** and **strPassword**.)



Encryption via TLS/SSL is not supported in projects running on Windows CE target systems.

Appendix: WinStudio-Internal Scripting Language

Returned value

Value	Description
0	Successful
1	Invalid format for strSMTP
2	Invalid format for strFrom
3	Invalid format for strPOP3
4	Invalid format for strUser
5	Invalid format for strPassword
6	Invalid format for optNumTimeout
7	Incorrect number of parameters
8	Error getting host IP address (invalid POP3 server)
9	Error connecting to POP3 server
10	Error sending user name
11	Error sending password
12	SMTP server does not support selected authentication mode
13	Invalid SMTP user name
14	Authentication failed



The email configuration created by this function works only within the Windows process where the function was called.

For example, if a button object is placed in a screen and the object is set to call this function when it is pressed, the resulting email configuration works only on the client station where the screen is displayed and the button is pressed.

It does not work on any other client station nor on the server station as the project viewer running on the client station only exchanges data (i.e., changes in tag values) with the data server running on the server station. A server cannot directly call functions on another server. Only triggers can be used to call functions on other servers.

This also applies if the client station and the server station are the same physical device as project viewer and data server are two separate processes in Windows.

If an email configuration is to apply to the Background tasks of the project – e.g., to be able to send emails when alarms become active –, either the "Email Settings" dialog has to be used to configure default settings for the entire project OR this function, for example, with the [startup script, page 325](#), a [script group, page 324](#) or a [math worksheet, page 324](#).

Examples *CnfEmail - Examples*

```
CnfEmail("smtp.company.com", "Robert@company.com", "pop.company.com", "RobertH", "Shades556", 100)
```

```
CnfEmail("smtp.company.com:4455", "Robert@company.com", "pop.company.com:9900", "RobertH", "Shades556", 5, 1)
```

```
CnfEmail("195.11.22.33:4455", "Robert@company.com", "195.66.77.88:9900", "RobertH", "Shades556", 5, 2, "JohnS", "abcd1234")
```

19.23.3 GetStatusSendEmailExt

Returns the status of the last email sent using the "SendEmailExt" function.

Function	Group	Execution	Windows	Embedded	Thin Client
GetStatusSendEmailExt	Email	synchronous	supported	supported	supported

Syntax

```
GetStatusSendEmailExt({ | optTagName })
```

Inputs

optTagName

"Optional" tag causing the function to update its return value. This parameter is optional. However, it has to be used if this function is configured for an [object animation](#), [page 220](#) (e.g., Text Data Link, Position).

Returned value

Value	Description
-2	Incorrect INDMail.DLL library version
-1	Corrupted INDMail.DLL library
0	SendEmailExt function is not executed
1	Still sending last email. Cannot execute the SendEmailExt function
2	Last email was sent successfully. The SendEmailExt function can be executed again
3	There was an error sending the last email. Execute the SendEmailExt function again

Examples

Tag name	Expression
Tag	GetStatusSendEmailExt(Second)
Tag	GetStatusSendEmailExt()

19.23.4 SendEmail

This function sends an email to a receiver (without attached files).

Function	Group	Execution	Windows	Embedded	Thin Client
SendEmail	Email	synchronous	supported	supported	supported

Syntax

```
SendEmail( strSubject, strMessage, strTo )
```

Inputs

strSubject

The subject of the email.

strMessage

The message text of the email, up to 255 characters long.

strTo

The email address of the recipient.

Appendix: WinStudio-Internal Scripting Language

Returned value

Value	Description
0	Successful
1	Invalid format for strSubject
2	Invalid format for strMessage
3	Invalid format for strTo
4	Incorrect number of parameters
5	Start socket error
6	Error getting host IP address (i.e., invalid SMTP server)
7	Error connecting to SMTP server
8	Error sending HELO command (i.e., initialization)
9	Error sending MAIL command (i.e., the "From" address)
10	Error sending RCPT command (i.e., the "To" address)
11	Error sending DATA (i.e., message text)
12	Error sending SMTP authentication command
13	Invalid user name
14	Invalid password



Before any email can be sent, the email settings of the project have to be configured first, either by using the **Email Settings** dialog or by calling the **CnfEmail** function. Incorrect configurations are indicated by the returned value (see above).

The function **SendEmail** cannot be used when encryption via TLS/SSL is enabled or to send an email that contains Unicode characters. For this purpose, the **SendEmailExt** function is available.

Examples

SendEmail - Examples

```
SendEmail( "Hi!", "How are you?", "rogers@pnd.net" )
```

```
SendEmail( statusSummary, statusDetail, adminAddress )
```

19.23.5 SendEmailExt

Sends an email with attached files.

Function	Group	Execution	Windows	Embedded	Thin Client
SendEmailExt	Email	asynchronous	supported	supported	supported

Syntax

```
SendEmailExt( strSubject, strMessage, strTo, optStrCc, optStrBcc, optStrFile1, ... , optStrFileN )
```

Inputs

strSubject

The email subject (up to 255 characters).

Appendix: WinStudio-Internal Scripting Language

strMessage

The email message (up to 255 characters).

strTo

The address of the recipient. More than one recipient can be specified. The addresses are separated by semicolons (;).

optStrCc

The recipient addresses for message copies. More than one recipient can be specified. The addresses are separated by semicolons (;).

This parameter is optional. If the following parameters are to be used a null string ("") can be specified.

optStrBcc

The recipient addresses for message "blind copies". More than one recipient can be specified. The addresses are separated by semicolons (;).

This parameter is optional. If the following parameters are to be used a null string ("") can be specified.

optStrFile (1-N)

Complete file paths and names of file attachments.

Returned value

Value	Description
-4	Some of the attached files could not be found
-3	Incorrect number of parameters (at least three parameters are required)
-2	Incorrect INDMail.DLL library version
-1	Corrupted INDMail.DLL library
0	Successful
1	The function cannot be executed as the last email has not been sent yet
2	Internal error



Before any email can be sent, the [CnfEmail, page 658](#) function has to be used to configure the email settings. Incorrect configurations are indicated by the returned value (see above).

Examples

Tag name	Expression
Tag	SendEmailExt("Subject", "Message", "Sam@universe.com", "", "", "C:\Projects eport.txt")
Tag	SendEmailExt("Subject", "Message", "David@Ohio.net", "Ted@Austin.com", "Bart@Springfield.gov", "C:\TechRef51.doc")

Appendix: WinStudio-Internal Scripting Language

19.24 Dial-up Functions

19.24.1 Dial-up Functions – General Information

These functions are used to configure computer modems (if applicable) and establish dial-up connections to other computers.



These functions are not supported by Windows 7.

Dial-up

Function	Execution	Windows PC	Windows CE	Thin Client
DialError , page 664	synchronous	supported (see note)	supported	supported
DialGetClientIP , page 670	synchronous	supported (see note)	supported	supported
DialGetServerIP , page 672	synchronous	supported (see note)	not supported	supported
DialStatus , page 673	synchronous	supported (see note)	supported	supported
DialUp , page 675	asynchronous	supported (see note)	supported	supported
DialUpToCE , page 676	asynchronous	supported (see note)	not supported	supported
FindAllDevices , page 678	synchronous	supported (see note)	supported	supported
FindModem , page 678	synchronous	supported (see note)	supported	supported
HangUp , page 679	synchronous	supported (see note)	supported	supported
PhoneDialUp , page 680	asynchronous	supported (see note)	not supported	supported
PhoneDisableListen , page 680	synchronous	supported (see note)	not supported	supported
PhoneEnableListen , page 681	synchronous	supported (see note)	not supported	supported
PhoneHangUp , page 682	asynchronous	supported (see note)	not supported	supported
PhoneStatus , page 682	synchronous	supported (see note)	not supported	supported

19.24.2 DialError

Returns the error codes regarding each connection.



This function is not supported by Windows 7.

Appendix: WinStudio-Internal Scripting Language

Function	Group	Execution	Windows	Embedded	Thin Client
DialError	Dial-up	synchronous	supported (see note)	supported	supported

Syntax

```
DialError(numType, strPhonebookEntryOrModem, "optError", optRefresh)
```

Inputs

numType

A numeric parameter specifying the content for the **strPhonebookEntryOrModem** parameter.

- 0: Phonebook name
- 1: Modem name
- 2: Direct connection name

strPhonebookEntryOrModem

The name of the phonebook, modem or direct connection to establish the connection. The **numType** parameter specifies which of these methods is used.

optError

"Optional" The name of the String tag receiving the error message.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

optRefresh

"Optional" tag causing the function to update its return value. This parameter is optional. However, it has to be configured if this function is to be used for an [object animation, page 220](#) (e.g., Text Data Link, Position).

Returned value

Value	Description
0	OK
-1	Error: INDRas.DLL not found
-2	Error: INDRas.DLL damaged
-3	Error: invalid number of parameters (minimum=2)
-4	Invalid value for the numType parameter (0 or 1)
-5	Phonebook or modem does not exist
600	An operation is pending
601	The port handle is invalid
602	The port is already open
603	Insufficient caller buffer
604	Incorrect information specified
605	Cannot set port information

Appendix: WinStudio-Internal Scripting Language

Value	Description
606	The port is not connected
607	The event is invalid
608	The device does not exist
609	The device type does not exist
610	Invalid buffer
611	The route is not available
612	The route is not allocated
613	Invalid compression specified
614	Out of buffer
615	The port could not be found
616	An asynchronous request is pending
617	The port or device is already disconnecting
618	The port is not open
619	The port is disconnected
620	There are no endpoints
621	Cannot open the phonebook file
622	Cannot load the phonebook file
623	Cannot open the phonebook file
624	Cannot write the phonebook file
625	Invalid information found in the phonebook file
626	Cannot load string
627	Cannot find key
628	The port was disconnected
629	The data link was terminated by the remote device
630	The port was disconnected due to hardware failure
631	The port was disconnected by the user
632	Incorrect structure size
633	The port is already in use or not configured for remote access dial out
634	The computer cannot be registered on the remote network
635	Unknown error
636	An incorrect device is connected to the port
637	The string could not be converted
638	The request has timed out
639	No asynchronous net available
640	A NetBIOS error occurred

Appendix: WinStudio-Internal Scripting Language

Value	Description
641	The server cannot allocate NetBIOS resources needed to support the client
642	One of the NetBIOS names is already registered on the remote network
643	A network adapter at the server failed
644	No network message pop-ups are received.
645	Internal authentication error
646	The account is not permitted to log on at this time of day
647	The account is disabled
648	The password has expired
649	The account does not have remote access rights
650	The Remote Access server is not responding
651	The modem (or other connecting device) has reported an error
652	Unrecognized response from the device
653	A macro required by the device was not found in the device .INF file section
654	A command or response in the device .INF file section refers to an undefined macro
655	The <message macro could not be found in the device .INF file section.
656	The <defaultoff macro in the device .INF file section contains an undefined macro
657	The device .INF file could not be opened
658	The device name in the device .INF or media .INI file is too long
659	The media .INF file refers to an unknown device name
660	The device .INF file contains no responses for the command
661	The device .INF file is missing a command
662	Attempted to set a macro not listed in the device .INF file section
663	The media .INI file refers to an unknown device type
664	Cannot allocate memory
665	The port is not configured for remote access
666	The modem (or other connecting device) does not work
667	Cannot read the media .INI file
668	The connection was disconnected
669	Invalid parameter used in the media .INI file
670	Cannot read the section name from the media .INI file
671	Cannot read the device type from the media .INI file
672	Cannot read the device name from the media .INI file
673	Cannot read the usage from the media .INI file

Appendix: WinStudio-Internal Scripting Language

Value	Description
674	Cannot read the maximum connection BPS rate from the media .INI file
675	Cannot read the maximum carrier BPS rate from the media .INI file
676	The line is busy
677	The call was answered by a person instead of a modem
678	No answer
679	No carrier found
680	No dial tone
681	General error reported by the device
682	ERROR_WRITING_SECTIONNAME
683	ERROR_WRITING_DEVICETYPE
684	ERROR_WRITING_DEVICENAME
685	ERROR_WRITING_MAXCONNECTBPS
686	ERROR_WRITING_MAXCARRIERBPS
687	ERROR_WRITING_USAGE
688	ERROR_WRITING_DEFAULTOFF
689	ERROR_READING_DEFAULTOFF
690	ERROR_EMPTY_INI_FILE
691	Access denied because of invalid user name and/or password on the domain
692	Hardware failure in port or attached device
693	ERROR_NOT_BINARY_MACRO
694	ERROR_DCB_NOT_FOUND
695	ERROR_STATE_MACHINES_NOT_STARTED
696	ERROR_STATE_MACHINES_ALREADY_STARTED
697	ERROR_PARTIAL_RESPONSE_LOOPING
698	A response key name in the device .INF file has an unexpected format
699	The device response caused a buffer overflow
700	The expanded command in the device .INF file is too long
701	The device moved to a BPS rate not supported by the COM driver
702	Unexpected device response received
703	The Application does not allow user interaction. The connection requires user interaction to run successfully
704	ERROR_BAD_CALLBACK_NUMBER
705	ERROR_INVALID_AUTH_STATE
706	ERROR_WRITING_INITBPS
707	X.25 diagnostic indication

Appendix: WinStudio-Internal Scripting Language

Value	Description
708	The account has expired
709	Error changing password on the domain. The password may be too short or too long or identical to a previously used password
710	Serial overrun errors were detected while communicating with the modem
711	RasMan initialization error. Check the event log
712	Biplex port initializing. Wait a few seconds and dial up again
713	No active ISDN lines are available
714	No ISDN channels are available for the call
715	Too many errors occurred due to poor phone line quality
716	The remote access IP configuration cannot be used
717	No IP addresses are available in the static pool of remote access IP addresses
718	Timeout waiting for a valid response from the remote PPP peer
719	PPP terminated by remote device
720	No PPP control protocols configured
721	Remote PPP peer is not responding
722	The PPP package is invalid
723	The phone number including area code and extension number is too long
724	The IPX protocol cannot dial out on the port because the device is an IPX router
725	The IPX protocol cannot dial up on the port because the IPX router is not installed
726	The IPX protocol cannot be used for dial out on more than one port at a time
727	Cannot access TCPCFG.DLL
728	IP adapter linked to remote access cannot be found
729	SLIP cannot be used unless the IP protocol is installed
730	Computer registration is not complete
731	The protocol is not configured
732	The PPP negotiation is not converging
733	The PPP control protocol for this network protocol is not available on the server
734	The PPP link control protocol was terminated
735	The requested address was rejected by the server
736	The remote computer terminated the control protocol
737	Loopback detected
738	The server has not assigned an address

Appendix: WinStudio-Internal Scripting Language

Value	Description
739	The authentication protocol required by the remote server cannot use the encrypted password reconnection of Windows NT, explicitly requesting password entry
740	Invalid TAPI configuration
741	The local computer does not support the required encryption type
742	The remote computer does not support the required encryption type
743	The remote computer requires encryption
744	The IPX network number assigned by the remote server cannot be used. Check the event log
745	ERROR_INVALID_SMM
746	ERROR_SMM_UNINITIALIZED
747	ERROR_NO_MAC_FOR_PORT
748	ERROR_SMM_TIMEOUT
749	ERROR_BAD_PHONE_NUMBER
750	ERROR_WRONG_MODULE
751	Invalid callback number. Only the characters "0" to "9", "T", "P", "W", "(", ")", "-", "@ and space are allowed in the number
752	A syntax error occurred while processing a script
753	The connection could not be disconnected because it was created by the Multi-Protocol Router

Examples

Tag name	Expression
Tag	DialError(0, "Office DialUp")
Tag	DialError(1, "USRobotics_SportsterFaxModem", "StatusMessage", second)
Tag	DialError(2, "DirectDial", "DialupError")

For further information, refer to: [FindModem\(\)](#), page 678

19.24.3 DialGetClientIP

Gets the IP address of a Remote Access Service (RAS) client station.



This function is not supported by Windows 7.

Function	Group	Execution	Windows	Embedded	Thin Client
DialGetClientIP	Dial-up	synchronous	supported (see note)	supported	supported

Syntax

```
DialGetClientIP({ numType | 0 | 1 | 2 }, strPhoneBookOrModem, "tagClientIP"{ | , optTagRefresh })
```

Appendix: WinStudio-Internal Scripting Language

Inputs

numType

The type of information specified by **strPhoneBookOrModem**: "0" is a phone-book name, "1" is a modem name and "2" is a direct connection name.

strPhoneBookOrModem

The name of the phonebook, modem or direct connection to establish the connection.

tagClientIP

The name of a String tag that is to receive the IP address.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

optTagRefresh

The name of a tag that triggers the function to update the returned value every time the value of the tag is changed.

This parameter is optional. However, it has to be specified if the function is to be used in an object animation, e.g., Text Data Link or Position.

Returned value

This function returns the following possible values:

Value	Description
-5	DialGetClientIP could not be found in IndRAS.dll
-4	Invalid type. Check numType
-3	Incorrect number of parameters
-2	DialStatus could not be found in IndRAS.dll
-1	IndRAS.dll was not loaded
n	Status code returned by DialStatus

The IP address is written to the tag specified by **tagClientIP**.



This function has to be executed on the RAS client station, **not** on the server station. (Essentially, the station is assigned its own IP address.)

Examples

Using the phonebook name "Office DialUp" and writing the resulting IP address to **ClientIPTag**:

DialGetClientIP(0, "Office DialUp", "ClientIPTag")

Using the modem name "USRobotics_SportsterFaxModem", writing the resulting IP address to **ClientIPAddress** and updating every second (i.e., every time the system tag **Second** changes):

DialGetClientIP(1, "USRobotics_SportsterFaxModem", "ClientIPAddress", Second)

Using the direct connection name "DirectDial" and writing the resulting IP address to **IPAdd**:

DialGetClientIP(2, "DirectDial", "IPAdd")

Appendix: WinStudio-Internal Scripting Language

19.24.4 DialGetServerIP

"DialGetServerIP" is an internal scripting function that gets the IP address of a Remote Access Service (RAS) server station.



This function is not supported by Windows 7.

Function	Group	Execution	Windows	Embedded	Thin Client
DialGetServerIP	Dial-up	synchronous	supported (see note)	not supported	supported

Syntax

```
DialGetServerIP({ numType | 0 | 1 | 2 }, strPhoneBookOrModem, "tagServerIP"{ | , optTagRefresh })
```

Inputs**numType**

The type of information specified by **strPhoneBookOrModem**: "0" is a phonebook name, "1" is a modem name and "2" is a direct connection name.

strPhoneBookOrModem

The name of the phonebook, modem or direct connection to establish the connection.

tagServerIP

The name of a String tag that is to receive the IP address.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

optTagRefresh

The name of a tag that triggers the function to update the returned value every time the value of the tag is changed.

This parameter is optional. However, it has to be specified if the function is to be used in an object animation, e.g., Text Data Link or Position.

Returned value

This function returns the following possible values:

Value	Description
-5	DialGetServerIP could not be found in IndRAS.dll
-4	Invalid type. Check numType
-3	Incorrect number of parameters
-2	DialStatus could not be found in IndRAS.dll
-1	IndRAS.dll was not loaded
n	Status code returned by DialStatus

The IP address is written to the tag specified by **tagServerIP**.

Appendix: WinStudio-Internal Scripting Language



This function has to be executed on the RAS client station, "not" on the server station. (Essentially, the station is assigned the IP address of the server to which it is connected.)

Examples

Using the phonebook name "Office DialUp" and writing the resulting IP address to **ServerIPTag**:

DialGetServerIP(0, "Office DialUp", "ServerIPTag")

Using the modem name "USRobotics_SportsterFaxModem", writing the resulting IP address to **ServerIPAddress** and updating every second (i.e., every time the system tag **Second** changes):

DialGetServerIP(1, "USRobotics_SportsterFaxModem", "ServerIPAddress", Second)

Using the direct connection name "DirectDial" and writing the resulting IP address to **IPAdd**:

DialGetServerIP(2, "DirectDial", "IPAdd")

19.24.5 DialStatus

Determines the status of the dial-up connection.



This function is not supported by Windows 7.

Function	Group	Execution	Windows	Embedded	Thin Client
DialStatus	Dial-up	synchronous	supported (see note)	supported	supported

Syntax

DialStatus(numType, strPhonebookEntryOrModem, "optStatus", optRefresh)

Inputs

numType

A numeric parameter specifying the content for the **strPhonebookEntryOrModem** parameter.

- 0: Phonebook name
- 1: Modem name
- 2: Direct connection name

strPhonebookEntryOrModem

The name of the phonebook, modem or direct connection to establish the connection. The **numType** parameter specifies which of these methods is used.

optStatus

"Optional" The name of the String tag receiving the status message.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Appendix: WinStudio-Internal Scripting Language

optRefresh

"Optional" tag causing the function to update its return value. This parameter is optional. However, it has to be configured if this function is to be used for an [object animation](#), [page 220](#) (e.g., Text Data Link, Position).

Returned value

Value	Description
-5	Phonebook or modem does not exist
-4	Invalid value for the numType parameter ("0" or "1")
-3	Error: Invalid number of parameters (minimum of 2)
-2	Error: INDRAS.DLL damaged
-1	Error: INDRAS.DLL not found
0	Opening the port...
1	Port was opened successfully
2	Connecting to the device...
3	The device has established a successful connection
4	All devices in the device chain have been connected successfully
5	Checking user name and password...
6	An authentication event has occurred
7	Another validation attempt with a new user was requested
8	Server has requested a callback number
9	The client has requested to change the password
10	Registering the computer on the network...
11	The connection speed calculation phase is starting...
12	An authentication request is being acknowledged
13	Reauthentication (after callback) is starting
14	The client has successfully completed authentication
15	The line is about to be disconnected for callback
16	Delaying to give the modem time to reset for callback
17	Waiting for an incoming call from the server
18	Projection result information is available
19	User authentication is being initiated or retried
20	Client has been called back and is about to resume authentication
21	Logging on to the network...
22	Subentry has been connected
23	Subentry has been disconnected
24	Terminal state supported by RASPHONE.EXE
25	Retry authentication state supported by RASPHONE.EXE
26	Callback state supported by RASPHONE.EXE

Appendix: WinStudio-Internal Scripting Language

Examples

Value	Description
27	Change password state supported by RASPHONE.EXE
8192	Successfully connected to remote server!
8193	Disconnected

Tag name	Expression
Tag	DialStatus(0, "Office DialUp")
Tag	DialStatus(1, "USRobotics_SportsterFaxModem", "StatusMessage", second)
Tag	DialStatus(2, "DirectDial", "DialupError")

For further information, refer to: [FindModem\(\)](#), page 678

19.24.6 DialUp

Establishes a dial-up connection.



This function is not supported by Windows 7.

Function	Group	Execution	Windows	Embedded	Thin Client
DialUp	Dial-up	asynchronous	supported (see note)	supported	supported

Syntax

DialUp(numType, strPhonebookEntryOrModem, strUserName, strPassword, optStrDomain, strPhoneNumber)

Inputs

numType

A numeric flag that specifies the content of the **strPhonebookEntryorModem** parameter.

- 0: Phonebook name
- 1: Modem name
- 2: Direct connection name

strPhonebookEntryOrModem

The name of the phonebook, modem or direct connection to establish the connection. The **numType** parameter specifies which of these methods is used.

strUserName

The user name for logging on.

strPassword

The password for logging on.

optStrDomain

"Optional" The domain name for logging in.

Appendix: WinStudio-Internal Scripting Language

strPhoneNumber

The phone number to be dialed (used only if parameter numType=1).

Returned value

Value	Description
0	OK: Dialing started
-1	Error: INDRAS.DLL not found
-2	Error: INDRAS.DLL damaged
-3	Error: Invalid number of parameters (minimum of 5)
-4	Invalid value for the numType parameter (0 or 1)
-5	Invalid value for the strPhonebookEntryOrModem parameter (String)
-6	Phonebook or modem does not exist
-7	Phonebook or modem in use
-8	Depending on the numType parameter: - If numType = 0: Could not read properties from phonebook. - If numType = 1: More than 1000 active connections simultaneously.
-9	Preliminary phonebook could not be created



The RAS server of the operating system automatically carries out dial-up for Windows PC stations.

Examples

Tag name	Expression
Tag	DialUp(0, "OfficeDialup", "Guest", "Password")
Tag	DialUp(1, "USRobotics_SportsterFaxModem", "HR12378", "HRPass", "15125554321")
Tag	DialUp(2, "DirectDial", "Rberton", "MyPassword", "156.48.25.0")

For further information, refer to: [FindModem\(\)](#), page 678

19.24.7 DialUpToCE

Runs the "DialUpToCE" program sending the information required for server call back by means of **CERasSvr.exe**.



This function is not supported by Windows 7.

Function	Group	Execution	Windows	Embedded	Thin Client
DialUpToCE	Dial-up	asynchronous	supported (see note)	not supported	supported

Syntax

DialUpToCE(numModem, strDialPhone, strMyNumber, strUser, strPassword, optStrDomain, optAutoDial, optAutoClose)

Appendix: WinStudio-Internal Scripting Language

Inputs

numModem

The modem used to dial to the Windows CE remote station.

strDialPhone

The phone number of the Windows CE remote station.

strMyNumber

The phone number sent to the Windows CE remote station. **CERasSvr.exe** calls back to this phone number.

strUser

The user name to be sent to the Windows CE remote station. **CERasSvr.exe** uses this name to connect to the Windows XP/Vista/7 computer after calling back.

strPassword

The password to be sent to the Windows CE remote station. **CERasSvr.exe** uses this password to connect to the Windows XP/Vista/7 computer after calling back.

optStrDomain

"Optional" The domain name for logging in.

optAutoDial

"Optional" tag with the following possible values:

- 1: Triggers the **DialUpToCE** connection automatically if the function is executed
- 0: Requests confirmation before triggering the **DialUpToCE** connection automatically if the function is executed

optAutoClose

"Optional" tag with the following possible values:

- 1: Closes the **DialUpToCE** dialog automatically after dialing the Windows CE remote station
- 0: Keeps open the **DialUpToCE** dialog

Returned value

Value	Description
0	Failed, unable to call "DialUpToCE".
1	Successful, "DialUpToCE" executed.

Appendix: WinStudio-Internal Scripting Language



The **DialUpToCE** program was developed to dial up a remote Windows CE station. As Windows CE v.3.00 does not provide a RAS server, prior to answering a call the **CERasSvr.exe** program and a call back have to be executed on the Windows XP/Vista/7 computer using parameters sent by the **DialUpToCE** function. The RAS server service has to be configured on the Windows XP/Vista/7 computer to answer the call back from the Windows Embedded device and set the TCP/IP connection.

Examples

Tag name	Expression
Tag	DialUpToCE(0, "12344321", "98765432", "Administrator", "MyPass")
Tag	DialUpToCE(0, "12344321", "98765432", "Administrator", "MyPass", "", 1, 1)

19.24.8 FindAllDevices

Returns the list of all available modems and direct connection interfaces (COM ports) on the local station.



This function is not supported by Windows 7.

Function	Group	Execution	Windows	Embedded	Thin Client
FindAllDevices	Dial-up	synchronous	supported (see note)	supported	supported

Syntax

```
FindAllDevices( "tagArray" )
```

Inputs

tagArray

Name of a String array tag receiving the list of available modems and direct connection interfaces.

Returned value

Returns the number of modems and/or interfaces found.

Examples

Tag name	Expression
Tag	FindAllDevices("SerialConnections[1]")

19.24.9 FindModem

Returns a list of all available modems in the local station.



This function is not supported by Windows 7.

Function	Group	Execution	Windows	Embedded	Thin Client
FindModem	Dial-up	synchronous	supported (see note)	supported	supported

Syntax

```
FindModem( "tagArray" )
```

Appendix: WinStudio-Internal Scripting Language

Inputs

tagArray

Name of a String array tag receiving the list of available modems.

Returned value

Returns the number of modems found.



This function can be used to get the names of the serial interface for a dial-up connection via modem. The determined information can be used for the "strPhonebookEntryOrModem" parameter in the [DialError](#), page 664, [DialStatus](#), page 673, [DialUp](#), page 675 and [HangUp](#), page 679 function.

Examples

Tag name	Expression
Tag	FindModem("Modems[1]")

19.24.10 HangUp

Terminates a dial-up connection.



This function is not supported by Windows 7.

Function	Group	Execution	Windows	Embedded	Thin Client
HangUp	Dial-up	synchronous	supported (see note)	not supported	supported

Syntax

HangUp(numType, strPhonebookEntryOrModem)

Inputs

numType

A numeric flag specifying the meaning of the **strPhonebookEntryOrModem** parameter.

- 0: Phonebook name
- 1: Modem name
- 2: Direct connection name

strPhonebookEntryOrModem

The name of the phonebook, modem or direct connection to establish the connection. The **numType** parameter specifies which of these methods is used.

Returned value

Value	Description
0	OK.
-1	Error: INDRAS.DLL not found
-2	Error: INDRAS.DLL damaged
-3	Invalid value for the numType parameter ("0" or "1")
-4	Phonebook or modem does not exist
-5	No configured modems exist

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	HangUp(0, "OfficeDialup")
Tag	HangUp(1, "USRobotics_SportsterFaxModem")
Tag	HangUp(2, "DirectDial", "Rberton", "MyPassword")

For further information, refer to: [FindModem\(\)](#), page 678

19.24.11 PhoneDialUp

Dials a phone number using the Telephony Application Program Interface (TAPI).



This function is not supported by Windows 7.

Function	Group	Execution	Windows	Embedded	Thin Client
PhoneDialUp	Dial-up	asynchronous	supported (see note)	not supported	supported

Syntax

```
PhoneDialUp( strPhoneNumber, optStrModemName )
```

Inputs

strPhoneNumber

Phone number to be called by the function.

optStrModemName

Name of the modem used to dial. If no modem is specified, the function uses the first modem found on the operating system.

Returned value

Value	Description
0	OK (dial triggered)
-1	Invalid number of parameters
-3	INDTAPI.DLL library not found
-4	PhoneDialUp () function not supported by the current INDTAPI.DLL library

Examples

Tag name	Expression
	PhoneDialUp ("512-123-4567")
	PhoneDialUp (StringPhoneNumberTag)
	PhoneDialUp (StringPhoneNumberTag, StringModemNameTag)

19.24.12 PhoneDisableListen

Terminates reception of the modem for incoming calls.



This function is not supported by Windows 7.

Appendix: WinStudio-Internal Scripting Language

Function	Group	Execution	Windows	Embedded	Thin Client
PhoneDisableListen	Dial-up	synchronous	supported (see note)	not supported	supported

Syntax

PhoneDisableListen(optStrModemName)

Inputs

optStrModemName

Name of the modem used to dial. If no modem is specified, the function uses the first modem found on the operating system.

Returned value

Value	Description
1	OK (no reception of incoming calls)
-1	INDTAPI.DLL library not found
-2	PhoneDisableListen () function not supported by the current INDTAPI.DLL library

Examples

Tag name	Expression
	PhoneDisableListen()
	PhoneDisableListen("Hayes Compatible Modem on COM1")
	PhoneDisableListen(StringModemNameTag)

For further information, refer to: [PhoneEnableListen\(\)](#), page 681

19.24.13 PhoneEnableListen

Resumes reception of the modem for incoming calls.



This function is not supported by Windows 7.

Function	Group	Execution	Windows	Embedded	Thin Client
PhoneEnableListen	Dial-up	synchronous	supported (see note)	not supported	supported

Syntax

PhoneEnableListen(optStrModemName)

Inputs

optStrModemName

Name of the modem used to dial. If no modem is specified, the function uses the first modem found on the operating system.

Returned value

Value	Description
1	OK (reception of incoming calls)
0	Error executing the PhoneEnableListen() function
-1	INDTAPI.DLL library not found
-2	PhoneEnableListen() function not supported by the current INDTAPI.DLL library

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
	PhoneEnableListen()
	PhoneEnableListen("Hayes Compatible Modem on COM1")
	PhoneEnableListen(StringModemNameTag)

For further information, refer to: [PhoneDisableListen\(\)](#), page 680

19.24.14 PhoneHangUp

Terminates a connection established with the "PhoneDialUp" function.



This function is not supported by Windows 7.

Function	Group	Execution	Windows	Embedded	Thin Client
PhoneHangUp	Dial-up	synchronous	supported (see note)	not supported	supported

Syntax

```
PhoneHangUp( optStrModemName )
```

Inputs

optStrModemName

Name of the modem used to dial. If no modem is specified, the function uses the first modem found on the operating system.

Returned value

Value	Description
1	OK (dial-up was interrupted)
-1	INDTAPI.DLL library not found
-2	PhoneHangUp() function not supported by the current INDTAPI.DLL library

Examples

Tag name	Expression
	PhoneHangUp()
	PhoneHangUp("Hayes Compatible Modem on COM1")
	PhoneHangUp(StringModemNameTag)

19.24.15 PhoneStatus

Checks the status of the current connections.



This function is not supported by Windows 7.

Function	Group	Execution	Windows	Embedded	Thin Client
PhoneStatus	Dial-up	synchronous	supported (see note)	not supported	supported

Syntax

```
PhoneStatus ( "strStatus", optStrModemName )
```

Appendix: WinStudio-Internal Scripting Language

Inputs

strStatus

Name of the tag receiving the status description text

optStrModemName

Name of the modem used to dial. If no modem is specified, the function uses the first modem found on the operating system.

Returned value

Value	Description
0	Ready to make a call
1	Call was terminated
2	Line ringing
3	Dial tone
4	Dialing call
5	Call in process
6	Callback
7	Line is busy
8	Line is idle
9	Disconnected

Examples

Tag name	Expression
Tag Status Code	PhoneStatus("String Tag Status")
Tag Status Code	PhoneStatus("String Tag Status", "Hayes Compatible Modem on COM1")
Tag Status Code	PhoneStatus("String Tag Status", "StringModemNameTag")

19.25 ActiveX and .NET control functions

19.25.1 ActiveX and .NET Control Functions – General Information

These functions are used to execute functions in ActiveX and .NET objects integrated in WinStudio screens and to retrieve and change the properties of these objects.

ActiveX Control and .NET Control

Function	Execution	Windows PC	Windows CE	Thin Client
XGet, page 684	asynchronous	supported	ActiveX controls only	supported
XRun, page 684	asynchronous	supported	ActiveX controls only	supported
XSet, page 685	asynchronous	supported	ActiveX controls only	supported

Appendix: WinStudio-Internal Scripting Language

19.25.2 XGet

This function gets the current value of a property of an ActiveX and .NET object.

Function	Group	Execution	Windows	Embedded	Thin Client
XGet	ActiveX and .NET Control	asynchronous	supported	ActiveX objects only (see notes)	supported

Syntax

```
XGet ( strName , strProperties )
```

Inputs**strName**

The unique name of the ActiveX and .NET object as configured in the **Name** field of the "Object Properties" dialog.

strProperties

The name of the property the value of which is to be retrieved. Available properties are listed in the "Configuration" (for ActiveX objects) or "Members" (for .NET objects) dialog.

Returned value

Returns the value of the specified property.



This function cannot be used in [tasks, page 57](#) or in the [global procedures, page 716](#) script. Additionally, .NET Controls are not supported by Windows CE.

Examples

Getting the current color property value on the "ActXRec" ActiveX object:

```
XGet( "ActXRec", "Color" )
```

19.25.3 XRun

This function executes a method on an ActiveX or .NET object.

Function	Group	Execution	Windows	Embedded	Thin Client
XRun	ActiveX and .NET Control	asynchronous	supported	ActiveX objects only (see notes)	supported

Syntax

```
XRun ( strName , strMethod , Parameter1 ,ParameterN)
```

Inputs**strName**

The unique name of the ActiveX and .NET object as configured in the **Name** field of the "Object Properties" dialog.

strMethod

The method to be executed. Available methods are listed in the "Configuration" (for ActiveX objects) or "Members" (for .NET objects) dialog.

Appendix: WinStudio-Internal Scripting Language

Parameter(1...N)

Data of various data types that are required by the method to run. The number of parameters can range from 0 to 255 and depends on the specified method. The data types (e.g., Boolean, Integer, Real or String) of referring tags have to match the parameters of the method.

Returned value Returns the result of executing the method. Not all methods return results.



This function cannot be used in [tasks, page 57](#) or in the [global procedures, page 716](#) script. Additionally, .NET controls are not supported by Windows CE.

Examples Running the XPos method on the "ActXCir" ActiveX object with four original values for the method:

XRun("ActXCir", "XPos", FALSE, 12, 4.6, "This is my text.")

Running the XPos method on the "ActXCir" ActiveX Control with four referring tags for the method:

XRun("ActXCir", "XPos", TagA, TagB, TagC, TagD)

19.25.4 XSet

This function sets a property value on an ActiveX or .NET object.

Function	Group	Execution	Windows	Embedded	Thin Client
XSet	ActiveX and .NET Control	asynchronous	supported	ActiveX objects only (see notes)	supported

Syntax

XSet(strName, strProperties, Value)

Inputs

strName

The unique name of the ActiveX and .NET object as configured in the **Name** field of the **Object Properties** dialog.

strProperties

The property the value of which is to be set. Available properties are listed in the **Configuration** (for ActiveX objects) or **Members** (for .NET objects) dialog.

Value

Custom value replacing the preselected property in the ActiveX or .NET object.

Returned value This function does not return any value.

This function cannot be used in [tasks, page 57](#) or in the [global procedures, page 716](#) script. Additionally, .NET controls are not supported by Windows CE.

Examples Changing the "Display" property from "ActXDisplay" to "Status Normal":

XSet("ActXDisplay", "Display", "Status Normal")

Appendix: WinStudio-Internal Scripting Language

19.26 Event Logger Functions

19.26.1 Event Logger Functions – General Information

These functions are used to send events and comments to the event logger.

Event logger

Function	Execution	Windows PC	Windows CE	Thin Client
SendEvent , page 686	synchronous	supported	supported	supported

19.26.2 SendEvent

Sends an event to the event logger.

The function generates an entry in the event logger file. If the **Comment** option is enabled, the user is prompted to enter a comment after executing the SendEvent() function. This comment is saved in the event logger file.

Function	Group	Execution	Windows	Embedded	Thin Client
SendEvent	Event logger	synchronous	supported	supported	supported

Syntax

```
SendEvent( strEvent, optBoolFlag, optStrComment )
```

Inputs

strEvent

The text to be saved in the event logger.

optBoolFlag

The event has no comment if the parameter is not specified or 0 (zero). Otherwise, a comment is saved with the event.

optStrComment

Comment text to be entered into the event logger file if the function is triggered (if optBoolFlag =1). This parameter is optional. Without specified text, the user is prompted to enter a comment by a default dialog.

Returned value

Value	Description
0	Successful
1	Event logger is disabled in the Event Settings dialog
2	Event logger is enabled, however, custom messages are disabled in the Event Settings dialog



This function is synchronous. Therefore, the execution of the function is only completed after the event data (including the optional comment) is saved to the database file. It is recommended not to configure this function in a Background task (e.g., "Math" or "Scheduler") if the comment is not to be used or directly entered into the function (without dialog).

Examples

Appendix: WinStudio-Internal Scripting Language

Tag name	Expression
Tag	SendEvent("Valve Open") // Saves the event message.
Tag	SendEvent("Valve Open Oven No. " + OvenID) // Saves the event message concatenated with the value of the OvenID tag.
Tag	SendEvent("Valve Open" , 1) // Displays the dialog where the operator can type comments.
Tag	SendEvent("Valve Open" , 1 , TagComment) // Saves the event message with the comment configured in the TagComment tag.

19.27 FTP Functions

19.27.1 FTP Functions – General Information

These functions are used to configure the FTP settings and to upload and download files to/from a remote server.

FTP

Function	Execution	Windows PC	Windows CE	Thin Client
CnfFTP, page 687	synchronous	supported	supported	supported
FTPGet, page 688	asynchronous	supported	supported	supported
FTPPut, page 690	asynchronous	supported	supported	supported
FTPStatus, page 691	synchronous	supported	supported	supported

19.27.2 CnfFTP

This function configures the FTP settings used by other functions in the project that can transfer files, e.g., the "FTPGet" and "FTPPut" functions.

Function	Group	Execution	Windows	Embedded	Thin Client
CnfFTP	FTP	synchronous	supported	supported	supported

Syntax

```
CnfFTP(strServer, optStrUser, optStrPwn, optNumPassiveMode, optNumPort)
```

Inputs

strServer

The address of the FTP server.

optStrUser

The user name for the FTP account.

This parameter is optional. If no value is specified, the user name "anonymous" is used by default.

optStrPwd

The password for the FTP account.

This parameter is optional. If no value is specified, the password field is left blank.

Appendix: WinStudio-Internal Scripting Language

optNumPassiveMode

A numeric flag that specifies whether the passive FTP mode is enabled. (The passive FTP can be used to bypass some firewall configurations.)

Value	Description
0	Passive FTP mode disabled
1	Passive FTP mode enabled

This parameter is optional. If no value is specified, the passive FTP mode is disabled by default.

optNumPort

The TCP/IP port number.

This parameter is optional. If no value is specified, port 21 is used by default.

Returned value

Value	Description
0	Successful
-1	Invalid number of parameters
-2	Invalid server name
-3	Invalid user name



This function does not actually connect to the server. For this reason, these error codes do not indicate the quality of the connection. They only indicate whether the FTP settings have been successfully configured.



This function has to be called at least once to configure these settings before the [FTPGet, page 688](#) and [FTPPut, page 690](#) functions can be used to transfer files.

Example

Tag name	Expression
Tag	CnfFTP("ftp.mycompany.com", "admin", "12345", 1) // Configures the FTP server using passive mode.

19.27.3 FTPGet

Gets a file from a remote server and saves it on the local computer.

Function	Group	Execution	Windows	Embedded	Thin Client
FTPGet	FTP	asynchronous	supported	supported	supported

Syntax

```
FTPGet( strRemoteFile, strLocalFile, numTransferType, numOverwrite )
```


Appendix: WinStudio-Internal Scripting Language

Inputs

strRemoteFile

The full path and name of the desired file on the remote server with the syntax **"file path/file name.extension"**. Some FTP servers are case sensitive. For this reason, always use correct capitalization.

strLocalFile

Full path and name the file is to be saved to on the local computer with the syntax **"C:\file name.extension"**.

numTransferType

Numeric parameter to specify the file transfer type (refer to the table). This parameter is optional. If no value is specified, the transfer type "Unknown" (0) is used by default.

Value	Description
0	Unknown
1	ASCII
2	Binary
10	Unknown, without buffering
11	ASCII, without buffering
12	Binary, without buffering



It is recommended to use option "10". It automatically determines the format (ASCII or Binary) of the remote file and defines the transfer type accordingly. Additionally, it forces the runtime system to download the file from the actual FTP server instead of from an intermediate proxy or cache server.

numOverWrite

A numeric value that specifies whether the local file (specified by **strLocalFile**) may be overwritten if it already exists. This parameter is optional. The default value is 0.

Value	Description
0	Do not overwrite – an error is returned if the file already exists
1	Overwrite

Returned value

Value	Description
1	Failed to establish the FTP connection
0	Successful
-1	Invalid number of parameters
-2	Unknown system error
-3	Invalid remote file

Appendix: WinStudio-Internal Scripting Language

Value	Description
-4	Invalid local file
-5	Invalid transfer type



Before this function can be called, the FTP settings (i.e., server address and login) have to be configured using the [CnfFTP](#), [page 687](#) function.

As the function is executed asynchronously, the [FTPStatus](#), [page 691](#) function has to be used to check the result of the transfer.

Example

Tag name	Expression
Tag	FTPGet("\\Reports\040303.txt", "C:\Report.txt") // Retrieves the file 040303.txt in the Reports directory, from the FTP server that was previously specified by CnfFTP() , page 687 , it then saves the file locally at C:\Report.txt .

19.27.4 FTPPut

Transfers a file to a remote server.

Function	Group	Execution	Windows	Embedded	Thin Client
FTPPut	FTP	asynchronous	supported	supported	supported

Syntax

```
FTPPut( strLocalFile, strRemoteFile, numTransferType )
```

Inputs

strLocalFile

Fully qualified name of the local file (e.g., "**C:\file.extension**")

strRemoteFile

Fully qualified name of the remote file (e.g., "**/Folder/File.extension**"). It has to be observed that some FTP servers are case sensitive. This has to be considered when entering the file name.

numTransferType

- 0 - Unknown
- 1 - ASCII
- 2 - Binary

Default is 0.

Returned value

Value	Description
1	Failed to create FTP thread
0	Successful
-1	Invalid number of parameters
-2	Unknown system error

Appendix: WinStudio-Internal Scripting Language

Value	Description
-3	Invalid remote file
-4	Invalid local file
-5	Invalid transfer type



Before this function can be called, the FTP settings (i.e., server address and login) have to be configured using the [CnfFTP, page 687](#) function.

As the function is executed asynchronously, the [FTPStatus, page 691](#) function has to be used to check the result of the transfer.

Example

Tag name	Expression
Tag	FTPput("C:\Report.txt", "\Reports\040303.txt") // Retrieves the file C:\Report.txt with the name 040303.txt in the folder Reports in the FTP Server.

19.27.5 FTPStatus

This function returns the current status of a file transfer started with "FTPGet" or "FTPput".

Function	Group	Execution	Windows	Embedded	Thin Client
FTPStatus	FTP	synchronous	supported	supported	supported

Syntax

```
FTPStatus("strStatusTag")
```

Inputs

strStatusTag

Name of the String tag that is to receive the status description (in plain text) on function feedback.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

Value	Description
1	Transaction executed successfully
2	Resolving name
3	Name resolved
4	Connecting to server
5	Connected to server
6	Closing connection
7	Connection closed
8	Sending request

Appendix: WinStudio-Internal Scripting Language

9	Request sent
10	Response received
11	Intermediate response received
12	Response received
13	Request completed
0	No transaction is being executed
-2	Invalid opttagErrorDescription
-6	Error opening the connection (see status String for details)
-7	Error establishing the connection (see status String for details)
-8	Error receiving the file (see status String for details)
-9	Transfer pending

Example

Tag name	Expression
StatusCode	FTPStatus("StatusDescription") // Retrieves the status of a current transfer. The return code is stored in the StatusCode tag and the description in the StatusDescription tag.

19.28 Database/ERP Functions

19.28.1 Database/ERP Functions – General Information

These functions are used to interact with external databases and ERP systems using SQL-like commands.

DB/ERP

Function	Execution	Windows PC	Windows CE	Thin Client
DBCursorClose, page 693	synchronous	supported	supported	supported
DBCursorCurrentRow, page 695	synchronous	supported	supported	supported
DBCursorGetValue, page 696	synchronous	supported	supported	supported
DBCursorMoveTo, page 697	synchronous	supported	supported	supported
DBCursorNext, page 698	synchronous	supported	supported	supported
DBCursorOpen, page 699	synchronous	supported	supported	supported
DBCursorOpenSQL, page 700	synchronous	supported	supported	supported
DBCursorPrevious, page 702	synchronous	supported	supported	supported
DBCursorRowCount, page 703	synchronous	supported	supported	supported
DBDelete, page 703	synchronous	supported	supported	supported
DBExecute, page 704	synchronous	supported	supported	supported
DBInsert, page 706	synchronous	supported	supported	supported
DBSelect, page 707	synchronous	supported	supported	supported
DBUpdate, page 709	synchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Function	Execution	Windows PC	Windows CE	Thin Client
SyncAlarm, page 710	asynchronous	supported	supported	executed on the server
SyncAlarmStatus, page 711	synchronous	supported	supported	executed on the server
SyncEvent, page 711	asynchronous	supported	supported	executed on the server
SyncEventStatus, page 712	synchronous	supported	supported	executed on the server
SyncTrend, page 713	asynchronous	supported	supported	executed on the server
SyncTrendStatus, page 714	synchronous	supported	supported	executed on the server

19.28.2 DBCursorClose

Closes the cursor and releases the result set.



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Function	Group	Execution	Windows	Embedded	Thin Client
DBCursorClose	Database/ERP	synchronous	supported	supported	supported

Syntax

```
DBCursorClose( numCur, "optStrErrorTag" )
```

Inputs

numCur

The cursor handle for the result set. The cursor handle is returned by [DBCursorOpen, page 699](#) or [DBCursorOpenSQL, page 700](#).

optStrErrorTag

The name of a String tag that is to receive detailed error messages if errors occur during runtime.

This parameter is optional.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

In case of errors, a negative value is returned. Otherwise, 0 is returned.



If the cursor is closed, it is destroyed and cannot be used again. A new cursor has to be created using [DBCursorOpen, page 699](#) or [DBCursorOpenSQL, page 700](#).

Examples

As used in a math worksheet:

Appendix: WinStudio-Internal Scripting Language

Tag name	Expression
nError-Code	DBCursorClose(nCursor)

19.28.3 DBCursorColumnCount

Gets the total number of columns in a SQL result set.



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Function	Group	Execution	Windows	Embedded	Thin Client
DBCursorColumnCount	Database/ERP	synchronous	supported	supported	supported

Syntax

```
DBCursorColumnCount( numCur, "optStrErrorTag" )
```

Inputs

numCur

The cursor handle for the result set. The cursor handle is returned by [DBCursorOpen](#), page 699 or [DBCursorOpenSQL](#), page 700.

optStrErrorTag

The name of a String tag that is to receive detailed error messages if errors occur during runtime.

This parameter is optional.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

Returns the number of columns. In case of errors, a negative value is returned.



Also refer to [DBCursorRowCount](#), page 703.

Examples

As used in a math worksheet:

Tag name	Expression
nError-Code	DBCursorColumnCount(nCursor)

19.28.4 DBCursorColumnInfo

Gets information about a column in a SQL result set. The column is specified by a number instead of a name. For this reason, this function can be used to retrieve unknown column names.

Appendix: WinStudio-Internal Scripting Language



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Function	Group	Execution	Windows	Embedded	Thin Client
DBCursorColumnInfo	Database/ERP	synchronous	supported	supported	supported

Syntax

```
DBCursorColumnInfo(numCur, numColumn, numTypeInfo, "optStrErrorTag")
```

Inputs

numCur

The cursor handle for the result set. The cursor handle is returned by [DBCursorOpen](#), page 699 or [DBCursorOpenSQL](#), page 700.

numColumn

The number of columns information is to be obtained about. Please note that a result set may include only some of the columns in the original database table.

numTypeInfo

The type of information to be retrieved about the column:

Value	Description
0	Column name
1	Column data type

optStrErrorTag

The name of a String tag that is to receive detailed error messages if errors occur during runtime.

This parameter is optional.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

In case of errors, a negative value is returned. Otherwise, 0 is returned.

Examples

As used in a math worksheet:

Tag name	Expression
ErrorCode	DBCursorColumnInfo(nCursor, 2, 0) // Gets the column name of the second column in the result set.

19.28.5 DBCursorCurrentRow

Returns the row number of the current row (i.e., the cursor position) in a SQL result set.

Appendix: WinStudio-Internal Scripting Language



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Function	Group	Execution	Windows	Embedded	Thin Client
DBCursorCurrentRow	Database/ERP	synchronous	supported	supported	supported

Syntax

```
DBCursorCurrentRow( numCur, "optStrErrorTag" )
```

Inputs**numCur**

The cursor handle for the result set. The cursor handle is returned by [DBCursorOpen](#), page 699 or [DBCursorOpenSQL](#), page 700.

optStrErrorTag

The name of a String tag that is to receive detailed error messages if errors occur during runtime.

This parameter is optional.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

Returns the number of the current row. In case of errors, a negative value is returned.

Examples

As used in a math worksheet:

Tag name	Expression
nRow	DBCursorCurrentRow(nCursor)

19.28.6 DBCursorGetValue

Gets the value of the specified column of the current row (i.e., the cursor position) in a SQL result set.



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Function	Group	Execution	Windows	Embedded	Thin Client
DBCursorGetValue	Database/ERP	synchronous	supported	supported	supported

Syntax

```
DBCursorGetValue(numCur, strColumn, "optStrErrorTag")
```


Appendix: WinStudio-Internal Scripting Language

Inputs

numCur

The cursor handle for the result set. The cursor handle is returned by [DBCursorOpen](#), page 699 or [DBCursorOpenSQL](#), page 700.

strColumn

The name of the column.

optStrErrorTag

The name of a String tag that is to receive detailed error messages if errors occur during runtime.

This parameter is optional.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

Returns the value of the specified column. If the value is NULL or the cursor is invalid, an empty string with quality BAD is returned.

Examples

As used in a math worksheet:

Tag name	Expression
Tag	DBCursorGetValue(nCursor, "Column1")

19.28.7 DBCursorMoveTo

Moves the cursor to the specified row in a SQL result set and copies the values of that row to the mapped tags. If the specified row does not exist – i.e., if it is outside the range of the result set – the function returns an error code and does not change the mapped tags.



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Function	Group	Execution	Windows	Embedded	Thin Client
DBCursorMoveTo	Database/ERP	synchronous	supported	supported	supported

Syntax

DBCursorMoveTo(numCur, numRows, "optStrErrorTag")

Inputs

numCur

The cursor handle for the result set. The cursor handle is returned by [DBCursorOpen](#), page 699 or [DBCursorOpenSQL](#), page 700.

numRow

The row of the result set to which the cursor is to be moved.

Appendix: WinStudio-Internal Scripting Language

optStrErrorTag

The name of a String tag that is to receive detailed error messages if errors occur during runtime.

This parameter is optional.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value In case of errors, a negative value is returned. Otherwise, 0 is returned.

Examples As used in a math worksheet:

Tag name	Expression
nErrorCode	DBCursorMoveTo(nCursor, 4) // Moves the cursor to the fourth row of the result set and copies those values.

19.28.8 DBCursorNext

Moves the cursor to the next row in a SQL result set and copies the values of that row to the mapped tags. If there is no next row – i.e., if the current row is the last – the function returns an error code and does not change the mapped tags.



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Function	Group	Execution	Windows	Embedded	Thin Client
DBCursorNext	Database/ERP	synchronous	supported	supported	supported

Syntax

```
DBCursorNext( numCur, "optStrErrorTag" )
```

Inputs**numCur**

The cursor handle for the result set. The cursor handle is returned by [DBCursorOpen](#), page 699 or [DBCursorOpenSQL](#), page 700.

optStrErrorTag

The name of a String tag that is to receive detailed error messages if errors occur during runtime.

This parameter is optional.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value In case of errors, a negative value is returned. Otherwise, 0 is returned.

Examples As used in a math worksheet:

Appendix: WinStudio-Internal Scripting Language

Tag name	Expression
nError-Code	DBCursorNext(nCursor)

19.28.9 DBCursorOpen

Selects a set of rows and columns in a database table, initializes the cursor at the first row of the result set, copies the values of that row to mapped tags and returns a cursor handle that can be referenced by other DB/ERP functions.



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Function	Group	Execution	Windows	Embedded	Thin Client
DBCursorOpen	Database/ERP	synchronous	supported	supported	supported

Syntax

```
DBCursorOpen(strDBConn, strTable, optStrCondition, optStrColumns, optStrTags, optStrOrder, "optStrErrorTag")
```

Inputs

strDBConn

The name of the database connection. Connections are configured in the [Database/ERP, page 328](#) folder.

strTable

The name of the database table.

optStrCondition

A string that specifies which rows of the table are to be selected. This corresponds to the SQL WHERE clause and the string is to follow the same syntax.

This parameter is optional. If no rows are specified, all rows of the table are selected.

optStrColumns

A string specifying the columns of the table to be selected. This list of column names is to be comma-separated.

This parameter is optional. If no columns are specified, all columns of the table are selected.

optStrTags

A string specifying the project tags to which the columns are to be mapped. This list of tag names is to be comma-separated and in the same order as the columns specified by **optStrColumns**. If the cursor is moved through the result set, the values in the current row are copied to these tags.

This parameter is optional. If no tags are specified, no values are copied.

Appendix: WinStudio-Internal Scripting Language

optStrOrder

The order in which the rows are to be sorted. This corresponds to the SQL ORDER BY clause and the string is to follow the same syntax.

This parameter is optional. If no order is specified, the rows are left in the default order of the table.

optStrErrorTag

The name of a String tag that is to receive detailed error messages if errors occur during runtime.

This parameter is optional.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

Returns a numeric value representing the cursor handle. In case of errors, a negative value is returned.



This function corresponds to a SQL SELECT statement, except that it breaks the clauses of the statement into separate function parameters. Users who are familiar with SQL and want to create SELECT statements from scratch can use [DBCursorOpenSQL](#), [page 700](#) instead.

Also refer to [DBCursorClose](#), [page 693](#).

Examples As used in a math worksheet:

Tag name	Expression
nCursor	<pre>DBCursorOpen("DB1", "Table1", "Column1 > 3", "Column1, Column2", "Tag1, Tag2", "Column1, Column2 DESC", "TagError")</pre> // Opens Table1 of DB1 and selects all rows where Column1 has a value greater than 3. Column1 is mapped to Tag1, and Column2 is mapped to Tag2. Lines are ordered first by Column1, then by Column2, in descending order. Error messages are written to TagError.

19.28.10 DBCursorOpenSQL

Selects a set of rows and columns in a database table, initializes the cursor at the first row of the result set, copies the values of that row to mapped tags and returns a cursor handle that can be referenced by other DB/ERP functions. (This function corresponds to a SQL SELECT statement.)



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Function	Group	Execution	Windows	Embedded	Thin Client
DBCursorOpenSQL	Database/ERP	synchronous	supported	supported	supported

Syntax

```
DBCursorOpenSQL( strDBConn, strSQL, optStrTags, "optStrErrorTag" )
```

Appendix: WinStudio-Internal Scripting Language

Inputs

strDBConn

The name of the database connection. Connections are configured in the [Database/ERP, page 328](#) folder.

strSQL

A string as complete, syntactically correct SQL SELECT statement.



In SQL, curly brackets ("{}") are typically used to enclose an expression that has to be evaluated before the entire SQL statement is executed. For example:

```
SELECT * INTO inmates FROM OPENROWSET
('MSDASQL','Driver={Microsoft Text Driver (*.txt; *.csv)};
DEFAULTDIR=C:\;Extensions=CSV;
','SELECT * FROM flat.csv')
```

In "WinStudio", however, curly brackets are used to reference tags in text fields that are typically not evaluated (e.g., in the label of a Button object). If an SQL statement with this type of expression is given to **DBCursorOpenSQL**, the project tries to evaluate the expression as tag reference and the function fails. To provide an SQL statement in a way that the project can evaluate the expression properly, a new String tag is to be created with the text of the expression and referenced in the SQL statement. For example:

```
$AuxTag = "{Microsoft Text Driver (*.txt; *.csv)}"
$DBCursorOpenSQL( "inmates", "SELECT * INTO inmates
FROM
OPENROWSET ('MSDASQL','Driver={AuxTag};
DEFAULTDIR=C:\;Extensions=CSV;
','SELECT * FROM flat.csv')")
```

optStrTags

A string that lists the project tags to which the columns are to be mapped. This list of tag names is to be comma-separated and in the same order as the columns specified by the WHERE clause of **strSQL**. If the cursor is moved through the result set, the values in the current row are copied to these tags.

This parameter is optional. If no tags are specified, no values are copied.

optStrErrorTag

The name of a String tag that is to receive detailed error messages if errors occur during runtime.

This parameter is optional.

Appendix: WinStudio-Internal Scripting Language



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

Returns the cursor handle as numeric value. In case of errors, a negative value is returned.



Also refer to [DBCursorClose](#), page 693.

Examples

As used in a math worksheet:

Tag name	Expression
nCursor	DBCursorOpenSQL("DB1", "SELECT Column1, Column2 FROM Table1 WHERE Column1 > 3 ORDER BY Column1, Column2 DESC", "Tag1, Tag2")

19.28.11 DBCursorPrevious

Moves the cursor to the previous row of the result set and copies the values of that row to the mapped tags. If there is no previous row – i.e, if the current row is the first –, the function returns an error code and does not change the mapped tags.



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Function	Group	Execution	Windows	Embedded	Thin Client
DBCursorPrevious	Database/ERP	synchronous	supported	supported	supported

Syntax

```
DBCursorPrevious( numCur, "optStrErrorTag" )
```

Inputs**numCur**

The cursor handle for the result set. The cursor handle is returned by [DBCursorOpen](#), page 699 or [DBCursorOpenSQL](#), page 700.

optStrErrorTag

The name of a String tag that is to receive detailed error messages if errors occur during runtime.

This parameter is optional.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

On successful execution, the value 0 is returned. In case of errors, a negative value is returned.

Examples

As used in a math worksheet:

Appendix: WinStudio-Internal Scripting Language

Tag name	Expression
ErrorCode	DBCursorPrevious(nCursor)

19.28.12 DBCursorRowCount

Gets the total number of rows in an SQL result set.

Also refer to [DBCursorColumnCount\(\)](#), page 694.



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Function	Group	Execution	Windows	Embedded	Thin Client
DBCursorRow-Count	Database/ERP	synchronous	supported	supported	supported

Syntax

```
DBCursorRowCount( numCur, "optStrErrorTag" )
```

Inputs

numCur

The cursor handle for the result set. The cursor handle is returned by [DBCursorOpen](#), page 699 or [DBCursorOpenSQL](#), page 700.

optStrErrorTag

The name of a String tag that is to receive detailed error messages if errors occur during runtime.

This parameter is optional.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

Returns the number of rows. In case of errors, a negative value is returned.



Also refer to [DBCursorColumnCount](#), page 694.

Examples

As used in a math worksheet:

Tag name	Expression
nRow-Count	DBCursorRowCount(nCursor)

19.28.13 DBDelete

Deletes selected rows from a database table. (This function corresponds to an SQL DELETE statement.)



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Appendix: WinStudio-Internal Scripting Language

Function	Group	Execution	Windows	Embedded	Thin Client
DBDelete	Database/ERP	synchronous	supported	supported	supported

Syntax

```
DBDelete(strDBConn, strTable, strCondition, "optStrErrorTag")
```

Inputs**strDBConn**

The name of the database connection. Connections are configured in the [Database/ERP, page 328](#) folder.

strTable

The name of the database table.

strCondition

A string that specifies which rows of the table are to be selected. This corresponds to the SQL WHERE clause and the string is to follow the same syntax.

To delete all rows in the table, a single space (" ") is to be entered as condition statement.

optStrErrorTag

The name of a String tag that is to receive detailed error messages if errors occur during runtime.

This parameter is optional.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

Returns the number of rows deleted from the table. In case of errors, a negative value is returned.

Examples

As used in a math worksheet:

Tag name	Expression
nRowsDeleted	DBDelete("DB1", "Table1", "Column1 > 1000", "TagError") // Deletes all rows in Table1 where the value of Column1 is greater than 1000. The returned value (i.e., the number of rows deleted) is written to TagError .
Tag	DBDelete("DB1", "Table1", " ") // Deletes all rows of Table1.

19.28.14 DBExecute

"DBExecute" is an internal scripting function that executes a custom SQL statement on an external database. If the statement is a query (e.g., SELECT), the database values are copied to specified array tags.

Function	Group	Execution	Windows	Embedded	Thin Client
DBExecute	Database/ERP	synchronous	supported	supported	supported

Appendix: WinStudio-Internal Scripting Language

Syntax

```
DBExecute(strDBConn, strSQL{ | , optStrTags, optNumMaxRows{ | , optStrErrorTag })
```

Inputs

strDBConn

The name of the database connection. Connections are configured in the **Database/ERP** folder in the "Project Explorer".

strSQL

A complete, syntactically correct SQL statement.



In SQL, curly brackets ("{}") are typically used to enclose an expression that has to be evaluated before the entire SQL statement is executed. For example:

```
SELECT * INTO inmates FROM OPENROWSET
('MSDASQL','Driver={Microsoft Text Driver (*.txt; *.csv)};
DEFAULTDIR=C:\;Extensions=CSV','SELECT * FROM flat.csv')
```

In "WinStudio", however, curly brackets are used to reference project tags in text fields that are typically not evaluated (e.g., in the label of a Button object). If an SQL statement with this type of expression is given to "DBExecute", the project tries to evaluate the expression as tag reference and the function fails. To provide an SQL statement in a way that the project can evaluate the expression properly, a new String tag is to be created with the text of the expression and referenced in the SQL statement. For example:

```
AuxTag = "{Microsoft Text Driver (*.txt; *.csv)}"
DBExecute( "inmates", "SELECT * INTO inmates FROM
OPENROWSET ('MSDASQL','Driver={AuxTag};
DEFAULTDIR=C:\;Extensions=CSV;
','SELECT * FROM flat.csv')")
```

optStrTags

A comma-separated list of the names of array tags in the project to which the columns of an SQL SELECT result set is to be mapped. The database values are copied to these array tags with the first row of the result set being copied to array index 0. It has to be ensured that the arrays are sufficiently large to receive all rows of the result set.

This parameter is only required if **strSQL** contains an SQL SELECT statement. For all other types of statements, this parameter is ignored and does not need to be specified. However, if the syntax of the function has to be maintained to continue to **optStrErrorTag**, this parameter is to be assigned an empty string ("").

Appendix: WinStudio-Internal Scripting Language

optNumMaxRows

The maximum number of rows to be copied from an SQL SELECT result set. In most cases, to copy all of the rows, a number greater than the expected number of rows in the result set is to be specified.

This parameter is only required if **strSQL** contains an SQL SELECT statement. For all other types of statements, this parameter is ignored and does not need to be specified. However, if the syntax of the function has to be maintained in order to continue to **optStrErrorTag**, this parameter has to be assigned a value of 0.

optStrErrorTag

The name of a String tag that is to receive detailed error messages if errors occur during runtime.

This parameter is optional.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

This function returns the total number of rows affected by the SQL statement. In case of errors, a negative value is returned.

Please note that this is the value returned by the function itself. In case of an SQL SELECT statement, the database values are copied to the array tags specified by **optStrTags**.



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Examples*Program:*

```
DBExecute( "DB1", "INSERT INTO Table1(Column1,Column2) values(1,1)" )
```

```
DBExecute("DB1","SELECT max(Column1),max(Column2) FROM Table1","Tag1,Tag2",1,"TagError")
```

19.28.15 DBInsert

Inserts one new row into a database table. (This function corresponds to an SQL INSERT statement.)



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Function	Group	Execution	Windows	Embedded	Thin Client
DBInsert	Database/ERP	synchronous	supported	supported	supported

Syntax

```
DBInsert( strDBConn, strTable, strValues, optStrColumns, "optStrErrorTag" )
```

Appendix: WinStudio-Internal Scripting Language

Inputs

strDBConn

The name of the database connection. Connections are configured in the **Database/ERP** folder in the "Project Explorer".

strTable

The name of the database table.

strValues

A string with the values to be written in the new row. This list of values is to be comma-separated and String values have to be enclosed in quotation marks.

optStrColumns

A string with a list of the columns into which the values are to be written. This list of column names is to be comma-separated and in the same order as the values specified by **strValues**.

This parameter is optional. If no columns are specified, the values are written in the default column order of the database table.

optStrErrorTag

The name of a String tag that is to receive detailed error messages if errors occur during runtime.

This parameter is optional.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

In case of errors, a negative value is returned. Otherwise, 1 is returned.

Examples

As used in a math worksheet:

Tag name	Expression
nError-Code	DBInsert("DB1", "Table1", "1, 'one'", "Column1, Column2")

19.28.16 DBSelect

"DBSelect" is an internal scripting function that selects a result set from an external database (equivalent to an SQL SELECT statement), maps the columns to array tags in the project and copies the values from the result set to the array tags.

Function	Group	Execution	Windows	Embedded	Thin Client
DBSelect	Database/ERP	synchronous	supported	supported	supported

Syntax

```
DBSelect(strDBConn, strTable, strTags, strColumns, strCondition, strOrder{ | , optNumMaxRows{ | , optStrErrorTag})
```

Appendix: WinStudio-Internal Scripting Language

Inputs	strDBConn
	The name of the database connection. Connections are configured in the Database/ERP folder in the "Project Explorer".
	strTable
	The name of the database table the result set is to be selected from.
	strTags
	A comma-separated list of the names of array tags in the project the columns of the database table are to be mapped to. The database values are copied to these array tags with the first row of the result set being copied to index 0. It has to be ensured that the arrays are sufficiently large to receive all rows of the result set.
	strColumns
	A comma-separated list with the columns to be selected from the database table. The list order is to correspond to the list in strTags . To select all columns in the table in their original order, this parameter is to be assigned an empty string ("").
	strCondition
	A statement specifying which rows in the database table are to be selected. This corresponds to the SQL WHERE clause and has to follow the same syntax. To select all of the rows in the table, this parameter is to be assigned an empty string ("").
	strOrder
	A statement specifying the order the rows are to be sorted in. This corresponds to the SQL ORDER BY clause and has to follow the same syntax. To keep the rows in their original order, this parameter is to be assigned an empty string ("").
	optNumMaxRows
	The maximum number of rows to be copied. In most cases, to copy all of the rows, a number greater than the expected number of rows in the result set is to be specified. This is an optional parameter. If no value is specified, only the first row of the result set is copied.
	optStrErrorTag
	The name of a String tag that is to receive detailed error messages if errors occur during runtime. This parameter is optional.

Appendix: WinStudio-Internal Scripting Language



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value

This function returns the total number of rows in the result set. In case of errors, a negative value is returned.

It has to be observed that this is the value returned by the function itself. The database values are copied to the array tags specified by **strTags**.



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Examples Program:

```
DBSelect( "DB1", "Table1", "Array1,Array2", "Column1,Column2", "", "" )
```

```
DBSelect("DB1","Table1","Array1,Array2","Column1,Column2","Column2 < Column1","Column1",4,"TagError")
```

19.28.17 DBUpdate

Selects a result set and writes the same value to all rows of a specified column. (This function corresponds to an SQL UPDATE statement.)



This function emulates SQL (Structured Query Language) database operations. Before using this function, users have to familiarize themselves with the structure and execution of SQL commands.

Function	Group	Execution	Windows	Embedded	Thin Client
DBUpdate	Database/ERP	synchronous	supported	supported	supported

Syntax

```
DBUpdate( strDBConn, strTable, strValues, strColumns, optStrCondition, "optStrErrorTag" )
```

Inputs

strDBConn

The name of the database connection. Available connections are displayed in the "Database Connections" dialog that can be accessed via the context menu of the **Tasks ► External database** folder.

strTable

The name of the database table.

strValues

A string that lists the values to be written to the columns. This list of values is to be comma-separated and String values have to be enclosed in quotation marks.

strColumns

A string with a list of the columns into which the values are to be written. This list of column names is to be comma-separated and in the same order as the values specified by **strValues**.

Appendix: WinStudio-Internal Scripting Language

optStrCondition

A string that specifies which rows of the table are to be selected. This corresponds to the SQL WHERE clause and the string is to follow the same syntax.

This parameter is optional. If no rows are specified, all rows of the table are selected.

optStrErrorTag

The name of a String tag that is to receive detailed error messages if errors occur during runtime.

This parameter is optional.



The tag name has to be enclosed in quotation marks as illustrated in the syntax diagram. Otherwise, the project tries to get the value of the specified tag.

Returned value Returns the number of rows updated. In case of errors, a negative value is returned.

Examples As used in a math worksheet:

Tag name	Expression
Tag	DBUpdate("DB1", "Table1", ""X", "Column2", "Column1 = 1", "TagError") // In Table1 of DB1, for all rows where Column1 equals 1, writes "X" to Column2.

19.28.18 SyncAlarm

Synchronizes the alarm database.

Function	Group	Execution	Windows	Embedded	Thin Client
SyncAlarm	Database/ERP	asynchronous	supported	supported	executed on the server

Syntax

```
SyncAlarm( optStrStartDate, optStrEndDate )
```

Inputs**optStrStartDate**

String with the start date. If this parameter is not specified, the current date is used.

optStrEndDate

String with the end date. If this parameter is not specified, the function uses the same value as for the start date.

Returned value

Value	Description
1	Failed to start synchronization; the database is probably being synchronized
0	Successful
-1	Invalid group number

Appendix: WinStudio-Internal Scripting Language

Value	Description
-2	The format is not set to "Database"
-4	The start date specified is invalid
-5	The end date specified is invalid
-6	The start date is greater than the end date



This function is executed asynchronously to ensure that it does not return the result of the synchronization. To get this information, the [SyncAlarmStatus](#), page 711 function is to be used.

Examples

Tag name	Expression
Tag	SyncAlarm() // Synchronizes the database using the current date
Tag	SyncAlarm("10/20/2004") // Synchronizes the database only for the day 10/20/2004
Tag	SyncAlarm("10/20/2004", "10/28/2004") // Synchronizes the database from 10/20/2004 to 10/28/2004

19.28.19 SyncAlarmStatus

Returns the status of a previously called "SyncAlarm" function.

Function	Group	Execution	Windows	Embedded	Thin Client
SyncAlarmStatus	Database/ERP	synchronous	supported	supported	executed on the server

Syntax

SyncAlarmStatus()

This function does not have any parameters.

Returned value

Value	Description
3	Synchronization complete
2	Synchronization failed
1	Synchronization in progress
0	No synchronization is being executed
-1	The format is not set to "Database"

Examples

Tag name	Expression
Tag	SyncAlarmStatus()

19.28.20 SyncEvent

Synchronizes the event database.

Function	Group	Execution	Windows	Embedded	Thin Client
SyncEvent	Database/ERP	asynchronous	supported	supported	executed on the server

Appendix: WinStudio-Internal Scripting Language

Syntax

```
SyncEvent( optStrStartDate, optStrEndDate )
```

Inputs**optStrStartDate**

String with the start date. If this parameter is not specified, the current date is used.

optStrEndDate

String with the end date. If this parameter is not specified, the function uses the same value as for the start date.

Returned value

Value	Description
1	Failed to start synchronization; the database is probably being synchronized
0	Successful
-1	Invalid group number
-2	The format is not set to "Database"
-4	The start date specified is invalid
-5	The end date specified is invalid
-6	The start date is greater than the end date



This function is executed asynchronously to ensure that it does not return the result of the synchronization. To get this information, the [SyncEventStatus, page 712](#) function is to be used.

Examples

Tag name	Expression
Tag	SyncEvent() // Synchronizes the database using the current date
Tag	SyncEvent("10/20/2004") // Synchronizes the database only for the day 10/20/2004
Tag	SyncEvent("10/20/2004", "10/28/2004") // Synchronizes the database from 10/20/2004 to 10/28/2004

19.28.21 SyncEventStatus

Returns the status of a previously called "SyncEvent" function.

Function	Group	Execution	Windows	Embedded	Thin Client
SyncEventStatus	Database/ERP	synchronous	supported	supported	executed on the server

Syntax

```
SyncEventStatus()
```

This function does not have any parameters.

Appendix: WinStudio-Internal Scripting Language

Returned value

Value	Description
3	Synchronization complete
2	Synchronization failed
1	Synchronization in progress
0	No synchronization is being executed
-1	The format is not set to "Database"

Examples

Tag name	Expression
Tag	SyncEventStatus()

19.28.22 SyncTrend

Synchronizes the trend database.

Function	Group	Execution	Windows	Embedded	Thin Client
Avg	Database/ERP	asynchronous	supported	supported	executed on the server

Syntax

```
SyncTrend( numGroup, optStrStartDate, optStrEndDate )
```

Inputs

numGroup

Number of the trend group (number of the trend worksheet).

optStrStartDate

String with the start date. If this parameter is not specified, the current date is used.

optStrEndDate

String with the end date. If this parameter is not specified, the function uses the same value as for the start date.

Returned value

Value	Description
1	Failed to start synchronization; the database is probably being synchronized
0	Successful
-1	Invalid group number
-2	The format is not set to "Database"
-4	The start date specified is invalid
-5	The end date specified is invalid
-6	The start date is greater than the end date



This function is executed asynchronously to ensure that it does not return the result of the synchronization. To get this information, the [SyncTrendStatus, page 714](#) function is to be used.

Appendix: WinStudio-Internal Scripting Language

Examples

Tag name	Expression
Tag	SyncTrend(1) // Synchronizes the group 1 database using the current date
Tag	SyncTrend(1, "10/20/2004") // Synchronizes the group 1 database only for the day 10/20/2004
Tag	SyncTrend(1, "10/20/2004", "10/28/2004") // Synchronizes the group 1 database from 10/20/2004 to 10/28/2004

19.28.23 SyncTrendStatus

Returns the status of a previously called "SyncTrend" function.

Function	Group	Execution	Windows	Embedded	Thin Client
SyncTrendStatus	Database/ERP	synchronous	supported	supported	executed on the server

Syntax

SyncTrendStatus(numGroup)

Inputs

numGroup

Number of the trend group (trend worksheet)

Returned value

Value	Description
3	Synchronization complete
2	Synchronization failed
1	Synchronization in progress
0	No synchronization is being executed
-1	The format is not set to "Database"

Examples

Tag name	Expression
Tag	SyncTrendStatus(1)

20 Appendix: VBScript Language

20.1 Overview of VBScript

VBScript is a simple, standardized and flexible scripting language that enables implementing logics and algorithms within the project.

"WinStudio" implements Visual Basic Scripting Edition 5.5 or higher.

Advantages of the "VBScript" language:

- Syntax, operators and functions.
- The ability to create new variables and procedures (functions and/or sub-routines).
- Access to properties, methods and/or events from COM objects, including ActiveX control elements.
- The ability to execute logic operations in any platform that supports VBScript, including Microsoft Windows-based PCs (running as "WinStudio" server station), Microsoft Windows Embedded devices (via CEView) and Internet Explorer (via Thin Client).



In case of doubt whether the image loaded on the device supports VBScript, please contact the hardware manufacturer. Hardware manufacturers have to enable the support for VBScript on Windows CE devices to enable CEView to execute the scripts configured in the VBScript language on the device.

The purpose of this documentation is to provide an overview for the use of VBScript commands with "Rexroth WinStudio". Furthermore, it can be used as a quick reference for the most used features of the language. For a full description of the language as well as its interfaces and functions, please contact Microsoft. (When this documentation was created, the VBScript documentation could be accessed directly at the [Microsoft Developer Network](#). This link, however, is beyond our control and may change without notice.)

20.2 VBScript Interfaces in the Software

20.2.1 VBScript Interfaces in the Software - General Information

The following table provides a summary of the VBScript interfaces supported by "Rexroth WinStudio":

Interface	Range of procedures and variables	Execution	Functionality
Global procedures	Graphics and tasks	-	Declaration of procedures
Graphic script	Graphic script interface only	Server (Viewer) + Thin Clients	Declaration of variables, declaration of procedures, execution
Screen script	Screen in which the script is configured	Server (Viewer) + Thin Clients	Declaration of variables, declaration of procedures, execution
Command animation	Object in which the script is configured	Server (Viewer) + Thin Clients	Declaration of variables, execution
ActiveX events	Object in which the script is configured	Server (Viewer) + Thin Clients	Declaration of variables, execution

Appendix: VBScript Language

Interface	Range of procedures and variables	Execution	Functionality
Start script	All script sheets from tasks	Server (BGTask)	Declaration of variables, declaration of procedures, execution
Script groups	Script group only	Server (BGTask)	Declaration of variables, execution

The following figure illustrates the range of each VBScript interface and the order they are scanned by "WinStudio":

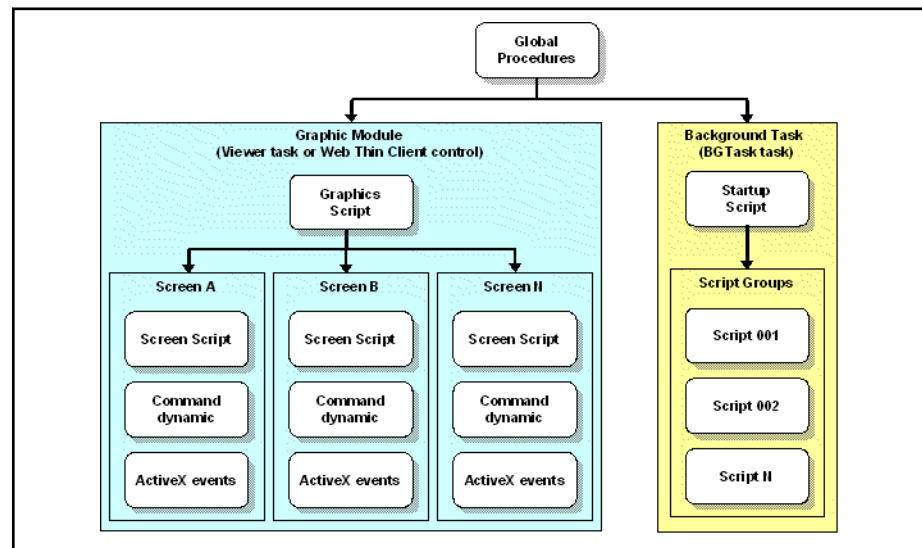


Fig.20-1: Range of VBScript interfaces

The figure illustrates that the global procedures are shared by the Graphic module and the Background task. However, the other VBScript interfaces are either from the Graphic module or from the Background task, and they do not share variables or procedures between them. They are independent of each other.

Although the graphic script is scanned by "WinStudio" before the screen scripts, procedures and variables declared in the graphic script interface are NOT available for any script interface configured on the screens. To implement procedures that have to be available to all screens, these have to be specified in the global procedures.

When writing the code into a VBScript interface, any tag from the "WinStudio" tags database or any function from the integrated scripting language can be accessed by applying the "\$" prefix to the tag/function name, as shown in the following examples:

Code in a VBScript interface

```

$Time 'Returns the value of the tag Time from the tags database
$MyTag 'Returns the value of the tag MyTag from the tags database
$Open("main") 'Executes the Open() built-in function to open the "main" screen

```

This way, VBScripts enable access to internal "WinStudio" functions, "WinStudio" tags, VBScript functions, VBScript variables, ActiveX properties, methods or events. "WinStudio" tags are shared by all modules from "WinStudio", including the Graphic module and the Background task.

20.2.2 Global Procedures

Worksheets in the "Global Procedures" folder can be used to create a library with [VBScript functions and sub-routines on page 744](#) whose functions can

Appendix: VBScript Language

be accessed by any other "WinStudio" scripting interface. The procedures declared here are never directly executed during runtime. They have to be called explicitly by another script.

NOTICE

"WinStudio" does not prevent declaration of two or more functions with the same name.

(This includes functions imported from external files. Refer to "Importing functions from an external file".) In this case, however, the project may behave unexpectedly during runtime. Make sure that all functions are named correctly and unambiguously.

Any number of worksheets for global procedures can be created and each worksheet can contain as many functions and sub-routines as required.

To create functions in a worksheet for global procedures, proceed as follows:

1. Open the default **Main Procedures** worksheet in folder **WinStudio ► Tasks ► Global Procedures**.

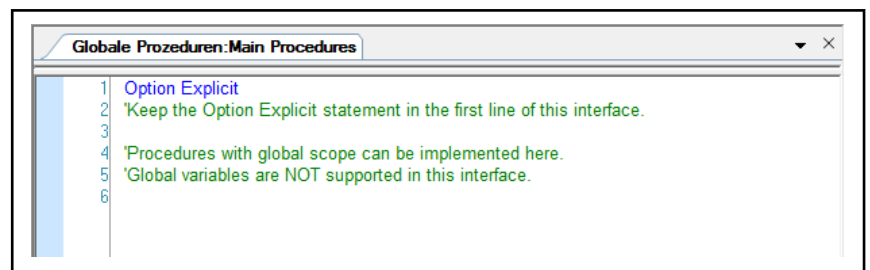


Fig.20-2: "Global Procedures - Main Procedures" worksheet

In this worksheet, any number of procedures can be declared. This worksheet is an integral part of any project and cannot be deleted. To structure procedures, however, additional worksheets can be created as follows.

2. Open the context menu of the **Global Procedures** folder and select "New Script"

Double-click on the inserted worksheet to open it in the editor.

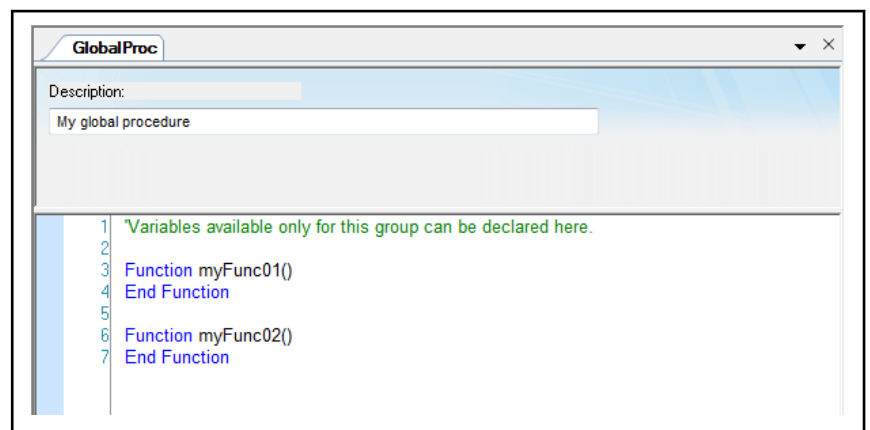


Fig.20-3: A newly inserted Global Procedures worksheet

3. Declare the functions and sub-routines in the interface.

For example:

```
'Procedures with global scope can be implemented here
'Global variables are NOT supported in this interface

Sub MyMessage(message)
```

Appendix: VBScript Language

```

    MsgBox(message,0)
End Sub

Function MyAdd(number1, number2)
    MyAdd = number1 + number2
    Call MyMessage("The sum is" & MyAdd & ".")
End Function

```



Local variables can be declared within each procedure, however, global variables cannot be declared in this interface. In most cases, tags are to be used instead.

4. Save changes.

Worksheet password protection

A password can be assigned to any of the global procedures worksheets to prevent them from being edited or analyzed by other users. To protect a worksheet, proceed as follows:

1. In the "Global Procedures" folder, right-click on the desired worksheet and select **Password Protection** from the context menu. The "Password Protection" dialog is displayed.
2. Enter the password into the **New Password** field.
3. Enter the password again into the **Confirm Password** field.
4. Click on **OK**.

Afterwards, the password is requested every time the worksheet is opened.

20.2.3 Graphic Module

Graphic Script

The "graphic script" worksheet is used to declare functions that are to be executed on certain events (see below) of the Graphic module. To open the graphic script, double click on the "Graphic Script" element in the "Scripts" folder.

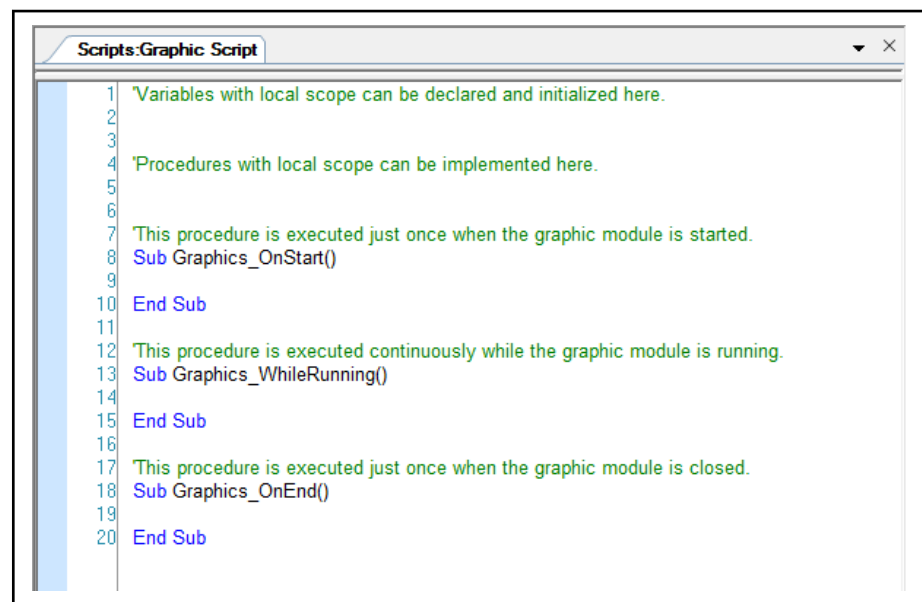


Fig.20-4: Project Explorer: Graphic script interface

This worksheet contains predefined functions to be executed on the following events:

- **Graphics_OnStart():** The code configured within this sub-routine is automatically executed once when the Graphic module is started. This inter-

Appendix: VBScript Language

face is useful for initializing variables or executing logics that have to be executed once when starting the runtime system.

- **Graphics_WhileRunning()**: The code configured within this sub-routine is automatically executed continuously while the Graphic module is running. The rate at which this sub-routine is called depends on the performance of the platform where the project is running.
- **Graphics_OnEnd()**: The code configured within this sub-routine is automatically executed once when the Graphic module is closed.
- **Graphics_OnScreenResize(width, height)**: The code configured within this sub-routine is automatically executed once when the screen resolution of the runtime station changes. The new height and width of the screen (in pixels) are passed to the sub-routine as parameters.



The **Graphics_OnScreenResize()** sub-routine is intended for projects running in CEView on Windows Embedded devices that can switch between portrait and landscape display modes. It is "not" supported for projects running on Windows PC or in the Thin Client.



Do not change the name of the predefined sub-routines. Otherwise, the system cannot execute them automatically.

Example:

```
'Variables with local scope can be declared and initialized here
Dim MyDate
MyDate = Date()
Dim MyValue
MyValue = 100

'Procedures with local scope can be implemented here
Function MyNewProcedure(nCount)
    MyNewProcedure = nCount + 1
End Function

Function AreaRec(side1, side2)
    AreaRec = side1 * side2
End Function

Sub CheckHiLimit(myValue, myHiLimit)
    If myValue > myHiLimit Then
        MsgBox("Value out of range")
    End If
End Sub

'This procedure is executed just once when the graphic module is started
Sub Graphics_OnStart()
    MsgBox("Welcome to the system!")
End Sub

'This procedure is executed continuously while the graphic module is running
Sub Graphics_WhileRunning()
    If $UserName = "Guest" Then
        $MyFlag = 0
    End If
End Sub

'This procedure is executed just once when the graphic module is closed
Sub Graphics_OnEnd()
    $LogOff()
End Sub
```

The following has to be observed when executing the graphic script:

On the server (where "Rexroth WinStudio" or CEView is running):

- The Graphic module is the Viewer task.

Appendix: VBScript Language

- The **Graphics_OnStart()** sub-routine is executed once on the server when the Viewer task is launched.
- The **Graphics_WhileRunning()** sub-routine keeps being executed on the server while the Viewer task is running. The **Graphics_OnEnd()** sub-routine is executed once on the server when the Viewer task is shut down.

On the Thin Client or Secure Viewer:

- The Graphic module is the ISSymbol control element.
- The **Graphics_OnStart()** sub-routine is executed once on the Thin Client station after logging in successfully.
- The **Graphics_WhileRunning()** sub-routine keeps being executed on the Thin Client station while the ISSymbol control element is hosted by the web browser.
- The **Graphics_OnStart()** sub-routine is executed once on the Thin Client station when the web browser is shut down (or when the ISSymbol control element is no longer hosted by the web browser).

The execution of the graphic script sub-routines on the server is completely independent of the execution on the Thin Client and Secure Viewer stations.

Calling graphic script procedures in other VBScript interfaces

The three predefined sub-routines are exclusively available to the Graphic module and are only executed on above mentioned events. In the graphic script, however, further functions can be declared that can be accessed by other interfaces (e.g., in screen scripts or command objects) by means of the **Graphics.procedure_name** syntax.

In the **MyNewProcedure** function declared in the example above, a Button object could be placed in the WinStudio screen and a command animation could be applied to it with the following line:

```
$NewTag = Graphics.MyNewProcedure($oldTag)
```

Screen Script

To edit the screen script of a WinStudio script, open it in the editor and select **Screen Script** from the context menu of the editor.

Screen scripts can be used to execute logics on the following events based on preconfigured sub-routines:

- **Screen_OnOpen()**: The code configured within this sub-routine is automatically executed once when the screen is opened.
- **Screen_WhileOpen()**: The code configured within this sub-routine is automatically executed continuously while the screen is open. The rate at which this sub-routine is called depends on the performance of the platform where the project is running.
- **Screen_OnClose()**: The code configured within this sub-routine is executed once when the screen is closed.
- **Screen_IsClosedByReplace()**: This procedure defines whether the screen is automatically closed if another screen is opened (this is the default behavior). If the function is assigned the value "0" or "FALSE", automatic closing is disabled. If the function is assigned a positive value (e.g., "1") or "TRUE" or if the procedure is not declared at all, automatic closing is enabled.

The variables and procedures declared in this interface are available to the VBScript interfaces of the screen where the screen script is configured.

NOTICE

Do "not" change the names of the preconfigured sub-routines described above.

Otherwise, they cannot be called by the system automatically.



- The execution of the screen script sub-routines on the server is completely independent of the execution on the Thin Client stations. In other words, these sub-routines are executed asynchronously.
- The procedures and/or variables declared in the screen script interface have local validity.
They can be called only from the specific screen on which they are declared.

Example:

```
'Variables available on this screen can be declared and initialized here
Dim Counter

'Procedures available on this screen can be implemented here
Function AreaCircle(radius)
    AreaCircle = Sqr(radius) * $Pi()
End Function

Sub CheckLoLimit (myValue, myLoLimit)
    If myValue < myLoLimit Then
        MsgBox("Value out of range")
    End If
End Sub

'This procedure determines whether the screen is automatically closed
Function Screen_IsClosedByReplace()
    Screen_IsClosedByReplace = $ReplaceModeTag
End Function

'This procedure is executed just once when this screen is open
Sub Screen_OnOpen()
    MsgBox("The screen was open!")
End Sub

'This procedure is executed continuously while this screen is open
Sub Screen_WhileOpen()
    If Counter < 100 Then
        Counter = Counter + 1
    Else
        Counter = 0
    End If
    $SimulationTag = Counter
End Sub

'This procedure is executed just once when this screen is closed
Sub Screen_OnClose()
    MsgBox("The screen will be closed!")
End Sub
```

Command Animation

To edit scripts in command animations, proceed as follows:

1. Select the object.
2. Click on the **Command on/off** icon in the "WinStudio dynamic function" toolbar.



Fig.20-5: Command animations icon

Appendix: VBScript Language

3. Double-click on the object. The "Object Properties" dialog for the command animation is displayed.

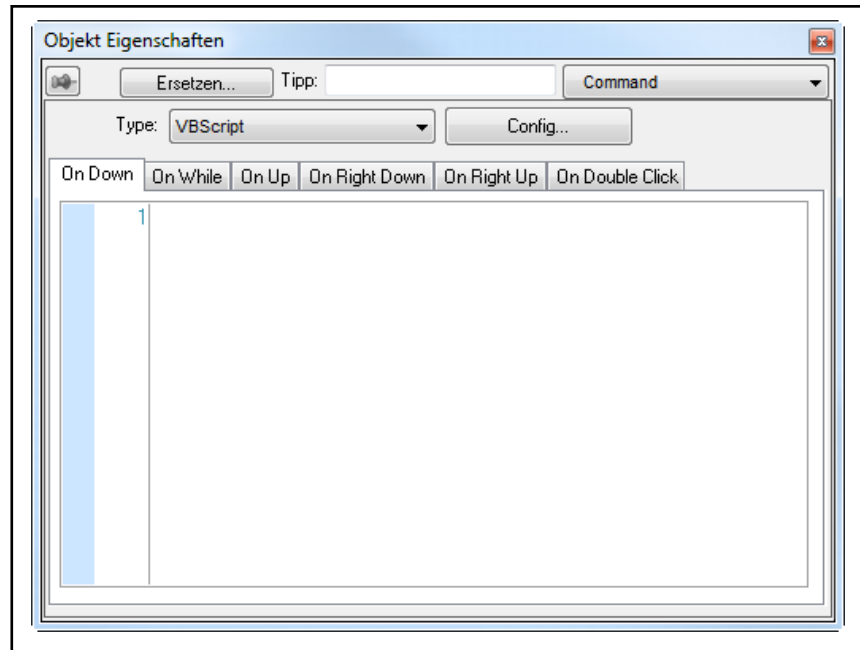


Fig.20-6: Object Properties dialog for the command animation

4. Select **VBScript** as **Type**.
5. To edit the script in a larger editor window or to set additional options, click on the **Config...** button.

Use this interface to execute functions when clicking on the object during runtime or pressing the shortcut configured for the object.

Variables declared in this interface are available for this interface only (local range). In other words, they are not available for any other object in the project. No procedures can be implemented in this interface. However, procedures can be called that are implemented in the global procedures or the screen script of the same screen in which the screen object can be found.



For further information, refer to [Command animation on page 220](#).

Example:

```
'The script below will be executed when the user clicks on the object
'where this animation is configured
$MyValue = InputBox("Please enter the new set-point", "Set-point")
```

ActiveX Events

To edit the ActiveX events interface, select the Script option from the Events tab of the ActiveX object in the screen.

1. Click on "ActiveX control" in the "WinStudio Screen Tools" tool window.

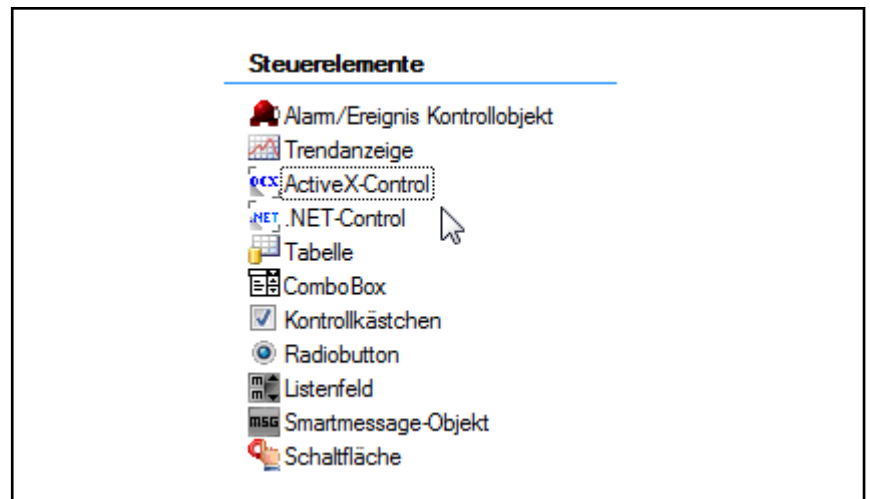


Fig.20-7: ActiveX control icon in the active objects toolbar

The "Insert ActiveX control" dialog is displayed.

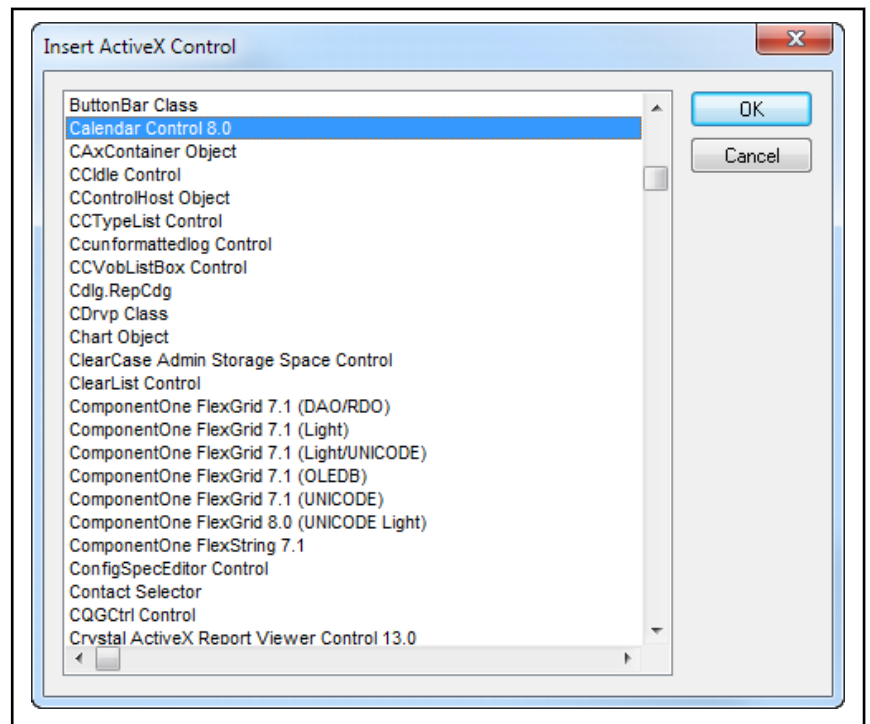


Fig.20-8: Insert ActiveX control dialog

2. Select the ActiveX control to be used and click on **OK**.
3. The object is created and inserted in the top-left corner of the screen. Double-click on the object to open the "Object Properties" dialog.

Appendix: VBScript Language

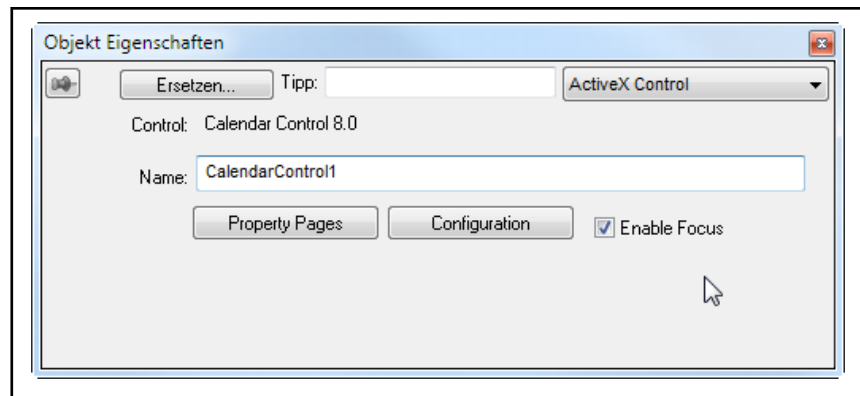


Fig.20-9: Object Properties dialog

4. Click on the **Configuration** button. The "Configuration" dialog is displayed. Click on the Event tab.

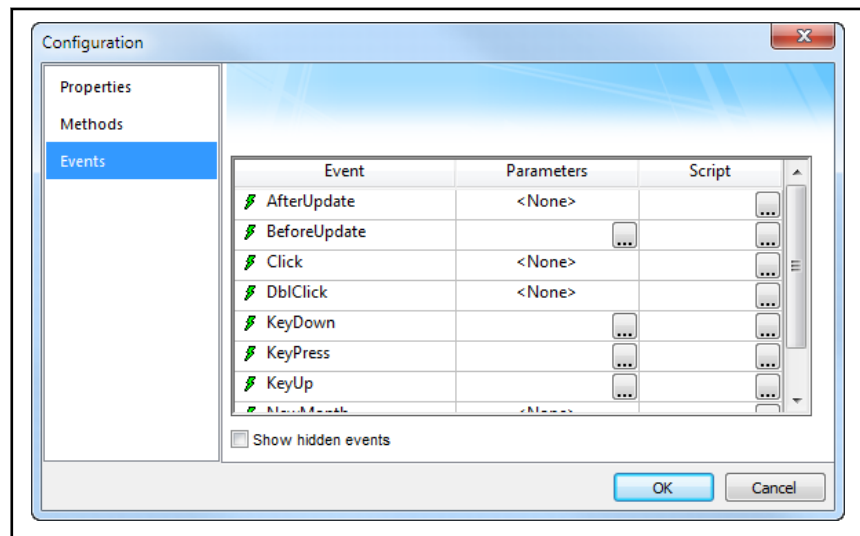


Fig.20-10: Event tab

5. Click on the ... button in the Script column.

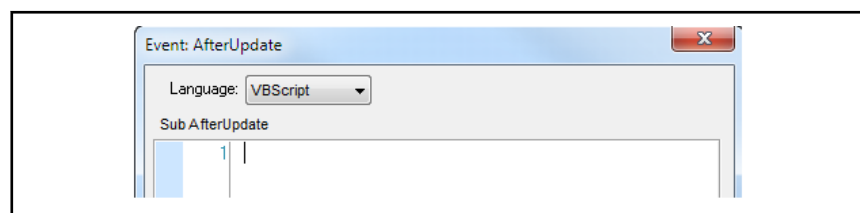


Fig.20-11: Script column

This interface is used to execute logics if an ActiveX object triggers an event. Variables declared in this interface are available for this interface only (local range). They are not included in other objects used in the project.

No procedures can be implemented in this interface. However, procedures can be called that are implemented in the global procedures or the screen script of the same WinStudio screen in which the ActiveX object can be found.



For further information, refer to [ActiveX control object](#), page 208.

Example:

```
'The script below will be executed when the Calendar Control ActiveX
'triggers its "AfterUpdate" event
$MyYear = CalendarControl1.Year
$MyMonth = CalendarControl1.Month
$MyDay = CalendarControl1.Day
```

20.2.4 Background Task

For further details, refer to:

- [Script worksheet Startup on page 325,](#)
- [Script worksheet on page 324.](#)

20.3 Language Reference

20.3.1 Language Reference – Overview

- [Constants, page 726](#)
- [Functions, page 734](#)
- [Errors, page 735](#)
- [Keywords, page 735](#)
- [Methods, page 733](#)
- [Objects and collections, page 730](#)
- [Operators, page 725](#)
- [Properties, page 730](#)
- [Statements, page 731](#)
- [System tags, page 738](#)

20.3.2 Operators

Arithmetic operators

Symbol	Name	Description
^	Exponentiation	Raises a number to the power of an exponent
-	Unary negation	Finds the difference between two numbers or indicates the negative value of a numeric expression
*	Multiplication	Multiplies two numbers
/	Division	Divides two numbers and returns a floating point result
\	Integer division	Divides two numbers and returns an integer result
Mod	Modulus arithmetic	Divides two numbers and returns only the remainder
+	Addition	Finds the sum of two numbers
-	Subtraction	Finds the difference between two numbers or indicates the negative value of a numeric expression
&	String concatenation	Forces string concatenation of two expressions

Appendix: VBScript Language

Comparison operators

Symbol	Name	Description
=	Equality	Comparison is True if the first expression is equal to the second expression
<>	Inequality	Comparison is True if the first expression is different from the second expression
<	Less than	Comparison is True if the first expression is less than the second expression
>	Greater than	Comparison is True if the first expression is greater than the second expression
<=	Less than or equal to	Comparison is True if the first expression is less than or equal to the second expression
>=	Greater than or equal to	Comparison is True if the first expression is greater than or equal to the second expression
Is	Object equivalence	Compares two object reference variables. Comparison is True if both object names refer to the same object

Logical operators

Symbol	Name	Description
Not	Logical negation	Performs logical negation on an expression
And	Logical conjunction	Performs a logical conjunction on two expressions
Or	Logical disjunction	Performs a logical disjunction on two expressions
Xor	Logical exclusion	Performs a logical exclusion on two expressions
Eqv	Logical equivalence	Performs a logical equivalence on two expressions
Imp	Logical implication	Performs a logical implication on two expressions

Assignment operators

Symbol	Name	Description
=	Assignment	Assigns a value to a variable or property

20.3.3 Constants**Color constants**

Constant	Value	Description
vbBlack	&h00	Black
vbRed	&hFF	Red
vbGreen	&hFF00	Green
vbYellow	&hFFFF	Yellow
vbBlue	&hFF0000	Blue
vbMagenta	&hFF00FF	Magenta

Appendix: VBScript Language

Constant	Value	Description
vbCyan	&hFFFF00	Cyan
vbWhite	&hFFFFFF	White

Comparison constants

Constant	Value	Description
vbBinaryCompare	0	Perform a binary comparison
vbTextCompare	1	Perform a textual comparison

Date & time constants

Constant	Value	Description
vbSunday	1	Sunday
vbMonday	2	Monday
vbTuesday	3	Tuesday
vbWednesday	4	Wednesday
vbThursday	5	Thursday
vbFriday	6	Friday
vbSaturday	7	Saturday
vbUseSystem-DayOfWeek	0	Uses the day of the week specified in the system settings for the first day of the week
vbFirstJan1	1	Uses the week in which January 1 occurs (default)
vbFirstFourDays	2	Uses the first week that has at least four days in the new year
vbFirstFullWeek	3	Uses the first full week of the year

Date format constants

Constant	Value	Description
vbGeneralDate	0	Display a date and/or time. For real numbers, date and time are displayed. If there is no fractional part, only a date is displayed. If there is no integer part, only the time is displayed. Date and time display depends on the system settings
vbLongDate	1	Displays a date using the long date format specified in the regional settings of the computer
vbShortDate	2	Displays a date using the short date format specified in the regional settings of the computer
vbLongTime	3	Displays a time using the long time format specified in the regional settings of the computer
vbShortTime	4	Displays a time using the short time format specified in the regional settings of the computer.

Appendix: VBScript Language

Miscellaneous constants

Constant	Value	Description
vbObjectError	-2147221504	User-defined error numbers have to be greater than this value

Window constants – Buttons and icons

Constant	Value	Description
vbOKOnly	0	Displays the OK button only
vbOKCancel	1	Displays OK and Cancel buttons
vbAbortRe-tryIgnore	2	Displays Abort, Retry and Ignore buttons
vbYesNoCancel	3	Displays Yes, No and Cancel buttons
vbYesNo	4	Displays Yes and No buttons
vbRetryCancel	5	Displays Retry and Cancel buttons
vbCritical	16	Displays a Critical Message icon
vbQuestion	32	Displays a Warning Query icon
vbExclamation	48	Displays a Warning Message icon
vbInformation	64	Displays an Information Message icon
vbDefaultBut-ton1	0	First button is the default
vbDefaultBut-ton2	256	Second button is the default
vbDefaultBut-ton3	512	Third button is the default
vbDefaultBut-ton4	768	Fourth button is the default
vbApplication-Modal	0	Application modal. The user has to respond to the message box before continuing to work in the current application
vbSystemModal	4096	System modal. On Win16 systems, all programs are suspended until the user responds to the message box. On Win32 systems, this constant provides a program modal message box that always remains on top of any other running programs

Box constants – Selected buttons

Constant	Value	Description
vbOK	1	OK button was clicked
vbCancel	2	Cancel button was clicked
vbAbort	3	Cancel button was clicked
vbRetry	4	Retry button was clicked
vbIgnore	5	Ignore button was clicked
vbYes	6	Yes button was clicked
vbNo	7	No button was clicked

Appendix: VBScript Language

String constants

Constant	Value	Description
vbCr	Chr(13)	Return
VbCrLf	Chr(13) & Chr(10)	Return...line feed combination
vbFormFeed	Chr(12)	Line feed; not useful in Microsoft Windows
vbLf	Chr(10)	Line feed
vbNewLine	Chr(13) & Chr(10) or	Platform-specific new line character; according to the platform
vbNullChar	Chr(0)	Characters with the value 0
vbNullString	String with the value 0	Not the same as a zero-length string (""); used for calling external procedures
vbTab	Chr(9)	Horizontal tab
vbVerticalTab	Chr(11)	Vertical tab; not useful in Microsoft Windows

Tristate constants

Constant	Value	Description
vbUseDefault	-2	Uses the default value from the regional settings of the computer
vbTrue	-1	TRUE
vbFalse	0	FALSE

VarType constants

Constant	Value	Description
vbEmpty	0	Uninitialized (default)
vbNull	1	Contains no valid data
vbInteger	2	Integer subtype
vbLong	3	Long subtype
vbSingle	4	Single subtype
vbDouble	5	Double subtype
vbCurrency	6	Currency subtype
vbDate	7	Date subtype
vbString	8	String subtype
vbObject	9	Object
vbError	10	Error subtype
vbBoolean	11	Boolean subtype
vbVariant	12	Variant (used only for arrays of variants)
vbDataObject	13	Data access object
vbDecimal	14	Decimal subtype

Appendix: VBScript Language

Constant	Value	Description
vbByte	17	Byte subtype
vbArray	8192	Array

20.3.4 Objects and Collections

Class object

The object created using the class statement. Provides access to the events of the class.

Debug object

An intrinsic global object that can send output to a script debugger like the Microsoft Script Debugger.

Err object

Contains information about runtime errors. Accepts the Raise and Clear methods for generating and clearing runtime errors.

Match object

Provides access to the write-protected properties of a regular expression match.

Matches collection

Collection of regular expression match objects.

Regular expression object (RegExp)

Provides simple regular expression support.

SubMatches collection

Collection of regular expression submatch strings.

20.3.5 Properties

Description

Returns or sets a descriptive string associated with an error.

FirstIndex

Returns the position in a search string where a match occurs.

Global

Sets or returns a Boolean value that indicates whether a pattern is to match all occurrences in an entire search string or just the first one.

HelpContext

Sets or returns a context ID for a topic in a help file.

Appendix: VBScript Language

HelpFile

Sets or returns a fully qualified path to a help file.

IgnoreCase

Sets or returns a Boolean value that indicates if a pattern search is case-sensitive or not.

Length

Sets or returns a Boolean value that indicates if a pattern search is case-sensitive or not.

Number

Returns or sets a numeric value specifying an error. Number is the default property of the Err object.

Pattern

Sets or returns the regular expression pattern being searched for.

Source

Returns or sets the name of the object or application that originally generated the error.

Value

Returns the value or text of a match found in a search string.

20.3.6 Statements

Call

Transfers the control to a sub or function procedure.

Class

Declares the name of a class and a definition of the variables, properties and methods that comprise the class.

Const

Declares constants for use instead of literal values.

Dim

Declares variables and allocates storage space.

Do...Loop

Repeats a block of statements while a condition is True or until a condition becomes True.

Appendix: VBScript Language

Erase

Reinitializes the elements of fixed-size arrays and removes the allocation of dynamic-array storage space.

Execute

Executes one or more specified statements.

ExecuteGlobal

Executes one or more specified statements in the global namespace of a script.

Exit

Exits a block of Do...Loop, For...Next, Function, or Sub code.

For Each...Next

Repeats a group of statements for each element in an array or collection.

For ... Next

Repeats a group of statements a specified number of times.

Function

Declares the name, arguments and code that form the body of a function procedure.

If...Then...Else

Conditionally executes a group of statements, depending on the value of an expression.

Option Explicit

Forces explicit declaration of all variables in a script.

Private

Declares private variables and allocates storage space. Declares a private variable in a class block.

Property Get

Declares the name, arguments and code in a class block that form the body of a property procedure that gets (returns) the value of a property.

Property Let

Declares the name, arguments and code in a class block that form the body of a property procedure that assigns (sets) the value of a property.

Appendix: VBScript Language

Property Set

Declares the name, arguments and code in a class block that form the body of a property procedure that sets a reference to an object.

Public

Declares public variables and allocates storage space. Declares a private variable in a class block.

Randomize

Initializes the random generator for numbers.

ReDim

Declares dynamic-array variables and allocates or reallocates storage space at procedure level.

Rem

Includes explanatory remarks in a program.

Select

Executes one of several groups of statements, depending on the value of an expression.

Set

Assigns an object reference to a variable or property or associates a procedure reference with an event.

Stop

Suspends execution.

Sub

Declares the name, arguments and code that form the body of a sub-procedure.

While

Executes a series of statements as long as a given condition is True.

With

Executes a series of statements on a single object.

Clear

Deletes all property settings of the Err object.

20.3.7 Methods

Appendix: VBScript Language

Execute

Executes a regular expression search with a specified String.

Raise

Generates a runtime error.

Replace

Replaces text found in a regular expression search.

Test

Executes a regular expression search with a specified String and returns a Boolean value that indicates if a pattern match was found.

Write

Sends Strings to the script debugger.

WriteLine

Sends Strings to the script debugger followed by a "New Line" character.

20.3.8 Functions

Function Names			
Abs	Array	Asc	Atn
CBool	CByte	CCur	CDate
CDBl	Chr	CInt	CLng
Conversions	Cos	CreateObject	CSng
CStr	Date	DateAdd	DateDiff
DatePart	DateSerial	DateValue	Day
Derived Math	Escape	Eval	Exp
Filter	FormatCurrency	FormatDateTime	FormatNumber
FormatPercent	GetLocale	GetObject	GetRef
Hex	Hour	InputBox	InStr
InStrRev	Int, Fix	IsArray	IsDate
IsEmpty	IsNull	IsNumeric	IsObject
Join	LBound	LCase	Left
Len	LoadPicture	Log	LTrim; RTrim; and Trim
Maths	Mid	Minute	Month
MonthName	MsgBox	Now	Oct
Replace	RGB	Right	Rnd
Round	ScriptEngine	ScriptEngineBuildVersion	ScriptEngineMajorVersion
ScriptEngineMinorVersion	Second	SetLocale	Sgn

Appendix: VBScript Language

Function Names			
Sin	Space	Split	Sqr
StrComp	Character string	StrReverse	Tan
Time	Timer	TimeSerial	TimeValue
TypeName	UBound	UCase	Unescape
VarType	Weekday	WeekdayName	Year

20.3.9 Keywords

Empty

The keyword Empty is used to indicate an uninitialized variable value. This does not correspond to Null.

False

The keyword False has a value equal to 0.

Nothing

The keyword Nothing in VBScript is used to disassociate an object variable from any actual object.

Null

The keyword Null is used to indicate that a variable contains no valid data. This does not correspond to Empty.

True

The keyword True has a value equal to -1.

20.3.10 Errors

VBScript runtime errors

Error number	Description
5	Invalid procedure call or argument
6	Overflow
7	Out of memory
9	Subscript out of range
10	This array is fixed or temporarily locked
11	Division by zero
13	Type mismatch
14	Out of string space
17	Cannot perform requested operation
28	Out of stack space
35	Sub or function not defined

Appendix: VBScript Language

Error number	Description
48	Error in loading DLL
51	Internal error
91	Object variable not set
92	"For" loop not initialized
94	Invalid use of Null
424	Object required
429	ActiveX component cannot create object
430	Class does not support automation
432	File name or class name not found during automation operation
438	Object does not support this property or method
445	Object does not support this action
447	Object does not support current local setting
448	Named argument not found
449	Argument not optional
450	Wrong number of arguments or invalid property assignment
451	Object not a collection
458	Variable uses an automation type not supported in VBScript
462	The remote server device does not exist or is not available
481	Invalid picture
500	Variable is undefined
502	Object not safe for scripting
503	Object not safe for initializing
504	Object not safe for creating
505	Invalid or unqualified reference
506	Class not defined
507	An exception occurred
5008	Invalid assignment
5017	Syntax error in regular expression
5018	Unexpected bracket characters
5019	Expected ']' in regular expression
5020	Expected ')' in regular expression
5021	Invalid range in character set

VBScript syntax errors

Error number	Description
1001	Out of memory
1002	Syntax error
1005	Expected '('
1006	Expected ')'
1010	Expected identifier
1011	Expected '='
1012	Expected 'If'
1013	Expected 'To'
1014	Expected 'End'
1015	Expected 'Function'
1016	Expected 'Sub'
1017	Expected 'Then'
1018	Expected 'Wend'
1019	Expected 'Loop'
1020	Expected 'Next'
1021	Expected 'Case'
1022	Expected 'Select'
1023	Expected expression
1024	Expected statement
1025	Expected end of statement
1026	Expected integer constant
1027	Expected 'While' or 'Until'
1028	Expected 'While', 'Until' or end of statement
1029	Expected 'With'
1030	Identifier too long
1037	Invalid use of 'Me' keyword
1038	'loop' without 'do'
1039	Invalid 'exit' statement
1040	Invalid 'for' loop control variable
1041	Name redefined
1042	Has to be first statement on the line
1044	Cannot use parentheses when calling a Sub
1045	Expected literal constant
1046	Expected 'In'
1047	Expected 'Class'

Appendix: VBScript Language

Error number	Description
1048	Has to be defined inside a class
1049	Expected Let or Set or Get in property declaration
1050	Expected 'Property'
1051	Number of arguments has to be consistent for properties specification
1052	Invalid multiple default properties/methods in a class
1053	No arguments to initialize or terminate class
1054	Property Set or Let requires at least one argument
1055	Unexpected 'Next'
1057	'Default' specification has to also specify 'Public'
1058	'Default' specification can only be on property Get

20.3.11 List of System Tags

List of system tags

Tag name	Data type	Description	Range
AckAll	Boolean	Toggle to "1" to acknowledge all alarms "1")"	Server
AckAlr	Boolean	Toggle to "1" to acknowledge the unacknowledged alarm with the highest priority; see "Alarm" below "1")"	Server
Alarm	Character string	The name of the unacknowledged alarm with the highest priority "1")"	Server
AnalogValue_	Real	A simulated analog value that steadily increases from "0" to "96" at the rate of 1.6/sec. If it reaches "96", it starts over at "0"	Local
Beep	Boolean	The current state of the audible alarm: "0" = OFF, "1" = ON "1")"	Server
BeepOff	Boolean	Toggle to "1" to disable the audible alarm "1")"	Server
BlinkFast	Boolean	Continuous, two-second blinking – one second on, one second off	Local
BlinkSlow	Boolean	Continuous, six-second blinking – three seconds on, three seconds off	Local
CrispDisplay	Integer	Legacy tags used by the CrispView custom interface. They are kept in "WinStudio" only for backward compatibility and should not be use in new projects. For more information, please contact technical support	Local
CrispInput	Character string		Local
CrispOutput	Character string		Local
Date	Character string	The current date on the local station presented as a string in MM/DD/YYYY format. For example, "05/15/2008". (Note: This may differ from the current date on the project server; see "ServerDate_" below)	Local
Day	Integer	The current day of the month, between "1" and "31"	Local
DayOfYear	Integer	The current day of the year, between "0" and "365"	Local
DigitalValue_	Boolean	A simulated digital value that alternates between "0" and "1" each second	Local
Goto	Character string	Goto, math function	Local

Appendix: VBScript Language

Tag name	Data type	Description	Range
GroupCNFHiLevel	Integer	End of the current security level range to CNF	Local
GroupCNFLoLevel	Integer	Start of current security level range to CNF	Local
GroupHiLevel	Integer	The maximum security level of the current user group "2)"	Local
GroupLoLevel	Integer	The minimum security level of the current user group "2)"	Local
GroupName	Character string	The name of the group to which the current user belongs "2)"	Local
Hint	Character string	The hint text for the current object. For further information, refer to Object Properties, page 164	Local
Hour	Integer	The current hour of the day, between "0" and "23"	Local
InputMaxRange	Real	The maximum value (Max) for the current tag or object; see "InputOutOfRange" below	Local
InputMinRange	Real	The minimum value (Min) for the current tag or object; see "InputOutOfRange" below	Local
InputOutOfRange	Boolean	This tag toggles to "1" when the user input is outside the Max/Min range of the current tag or object	Local
Label	Character string	Label, math function	Local
LastCodeChar_	Integer	Last code character in the Viewer (???)	Local
LptOff	Boolean	Toggle to "1" to disable the printing of alarms to the default printer "1)"	Local
Minute	Integer	The current minute of the hour, between "0" and "59"	Local
Month	Integer	The current month of the year, between "1" and "12"	Local
Next	Integer	Next, math function	Local
Reserved___1	Boolean	Reserved for future use	Local
Reserved___2	Boolean	Reserved for future use	Local
Reserved___8	Boolean	Reserved for future use	Local
Reserved___9	Boolean	Reserved for future use	Local
Reserved___10	Boolean	Reserved for future use	Local
Second	Integer	The current second of the minute, between "0" and "59"	Local
ServerDate_	Character string	The current date on the project server, presented as a string in MM/DD/YYYY format. For example, "05/15/2008".	Server
ServerTime_	Character string	The current time on the project server, presented as a string in HH:MM:SS format. For example, "23:45:03".	Server
Time	Character string	The current time on the local station, presented as a string in HH:MM:SS format. For example, "23:45:03". (Note: This may be different than the current time on the project server; see "ServerTime_" above.)	Local
Tomorrow	Integer	The next day of the month, between "1" and "31"	Local
UserName	Character string	The name of the user that is currently logged in "2)"	Local

Appendix: VBScript Language

Tag name	Data type	Description	Range
Weekday	Integer	The current day of the week on the local station, presented as an integer: "0" = Sunday, "1" = Monday, "2" = Tuesday, "3" = Wednesday, "4" = Thursday, "5" = Friday, "6" = Saturday	Local
Year	Integer	The current, four-digit year (e.g., "2008").	Local
Yesterday	Integer	The previous day of the month, between "1" and "31"	Local

"1)" For further information about configuring alarms, please refer to [Alarms folder, page 247](#).

"2)" For further information about configuring Users, Groups, and Security Levels, please see [Security folder, page 357](#).

20.4 Tips & Tricks

20.4.1 Tips & Tricks – Overview

- [VBScript editor IntelliSense, page 740](#)
- [VBScript compared to VBA, page 742](#)
- [Screen events, page 744](#)
- [MsgBox\(\) and InputBox\(\) functions, page 744](#)
- [Support for ActiveX objects, page 754](#)
- [Logical operator NOT, page 749](#)
- [Boolean variables, page 747](#)
- [Windows CE Support, page 755](#)
- [Scope and lifetime of variables, page 747](#)
- [Declaring variables, page 746](#)
- [Creating constants, page 746](#)
- [Precedence of VBScript operators, page 749](#)
- [Using conditional statements, page 750](#)
- [Looping through code, page 752](#)
- [VBScript procedures, page 744](#)

20.4.2 VBScript Editor IntelliSense

IntelliSense provides an array of options that make language references easily accessible. When coding, the Code Editor or Immediate Mode command window does not need to be left to perform searches on language elements. The context can be kept to find the required information, insert language elements directly into the code and even have IntelliSense complete typing.

IntelliSense comprises the following options...

List members

A list of valid members from class tags, fields from any tag, properties/methods from an ActiveX object or functions from the integrated scripting language can be displayed. Selecting from the list inserts the member into the code.

Type the \$ character on any VBScript interface to automatically open a list box with all tags available for the current project as well as all functions from the integrated scripting language.

Appendix: VBScript Language

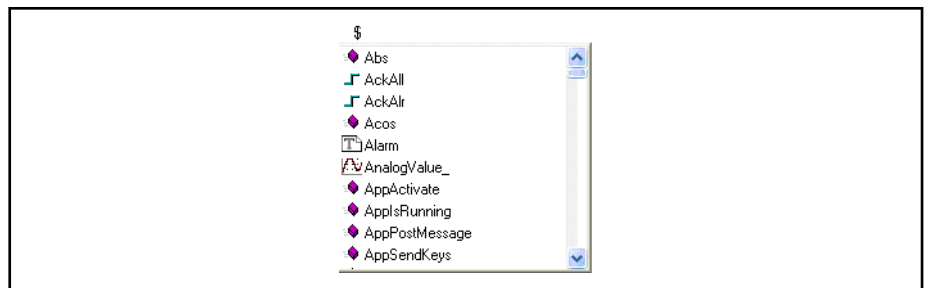


Fig.20-12: Opening the tag/function list box with the \$ character

By entering the name of a class tag followed by a point (.) into any VBScript interface, a list box with all members from the class tags is opened automatically:

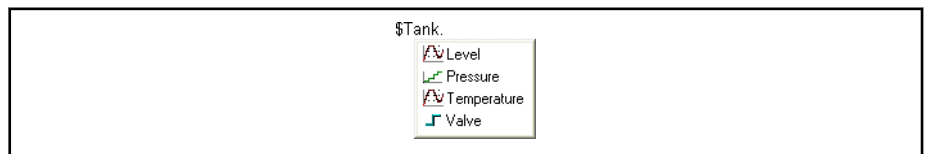


Fig.20-13: Opening the members list box with the "." character

By entering the name of a tag followed by a hyphen and a greater than character (->) into any VBScript interface, a list box with all fields available for this tag is opened automatically:

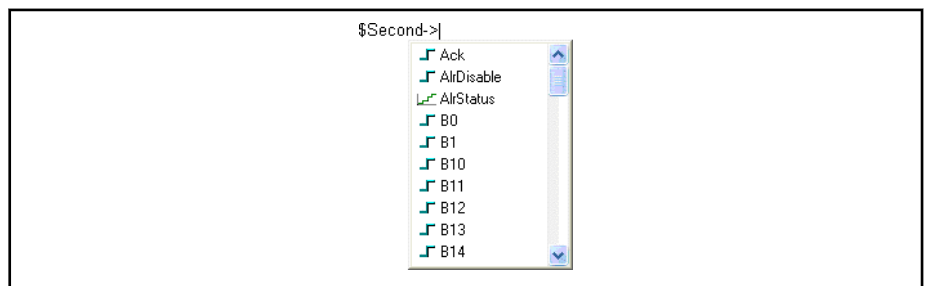


Fig.20-14: Opening the list box with all fields available for a tag with "->"

The elements are displayed in alphabetic order and each element has one of the following icons identifying its main type:

Symbol	Type
	Boolean tag
	Integer tag
	Real tag
	String tag
	Class tag
	Function from the integrated scripting language

By entering the name of an ActiveX control element followed by a point (.) into any VBScript interface from the screen inserted into the ActiveX object, a list box with all properties and methods of this object is automatically opened:

Appendix: VBScript Language

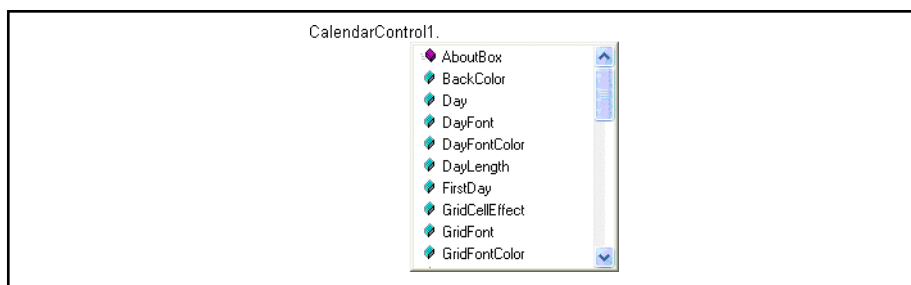


Fig.20-15: Opening the properties and methods list box with the "." character

The elements are displayed in alphabetic order and each element has one of the following icons identifying its main type:

Symbol	Type
	Property from the ActiveX object
	Method from the ActiveX object

Parameter Quick info

The Quick info option displays pop-up boxes with information about the functions from the integrated scripting language. The information includes all parameters supported by this function with the currently configured one in bold text.

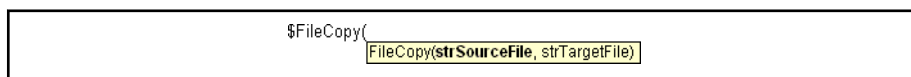


Fig.20-16: Parameter Quick info

Complete word

Complete word finishes a tag, member, field, function or ActiveX property/method name once enough unique characters for this term are entered. After entering the first few characters of the name, it can be completed automatically by pressing <Ctrl>+<Space>.

20.4.3 VBScript compared to VBA

While VBScript and Visual Basic for Applications (VBA) are similar and are both based on the Visual Basic standard language, there are advantages to using VBScript for "WinStudio" users:

- VBScript is supported by the Microsoft Windows Embedded operating system and VBA is not.
- VBScript brings active scripting to a wide variety of environments including web client scripting in Microsoft Internet Explorer. This prevents operations that may present risks for Thin Client users such as direct access to local files.
- VBScript was designed to be simple and easy to learn with some VBA standards simplified in VBScript to make it more straightforward. For example, in VBScript, users do not have to worry about the type of each variable when declaring them as VBScript automatically assumes the proper type for each variable.

The following table lists VBScript features that are not offered by VBA.

Category	Feature/Keyword
Declarations	Class
Miscellaneous	Eval
	Execute
Objects	RegExp

Appendix: VBScript Language

Category	Feature/Keyword
Script Engine Identification	ScriptEngine
	ScriptEngineBuildVersion
	ScriptEngineMajorVersion

The following table lists VBA features that are not offered by VBScript.

Category	Not Included Feature/Keyword
Array handling	Option Base
	Declaring arrays with lower limit <> 0
Collection	Add, Count, Item, Remove
	Access to collections using ! characters
Conditional compilation	#Const
	#If...Then...#Else
Control flow	DoEvents
	GoSub...Return, GoTo
	On Error GoTo
	On...GoSub, On...GoTo
	Line numbers, line labels
Conversion	CVar, CDate
	Str, Val
Data types	All intrinsic data types except Variant
	Type...End Type
Date/Time	Date statement, time statement
DDE	LinkExecute, LinkPoke, LinkRequest, LinkSend
Troubleshooting	Debug.Print
	End, Stop
Declaration	Declare (for declaring DLLs)
	Optional
	ParamArray
	Static
Error handling	Erl
	Error
	Resume, Resume Next
File Input/Output	All conventional basic files I/O
Financial	All financial functions
Object handling	TypeOf
Objects	Clipboard
	Collection

Appendix: VBScript Language

Category	Not Included Feature/Keyword
Operators	Like
Options	Deftype
	Option Base
	Option Compare
	Option Private Module
Select case	Expressions containing the "Is" keyword or any comparison operators
	Expressions containing a range of values using the "To" keyword
Strings	Fixed-length Strings
	LSet, RSet
	Mid Statement
	StrConv
Using objects	Collection access using !

20.4.4 Screen Events

In addition to the screen script, logics can be configured using the integrated scripting language for the On Open, While Open and On Close events for the screen (refer to the screen logic interface from the Screen Attributes dialog). For configuration of the screen script (VBScript language) and the screen logic (integrated scripting language), "WinStudio" respects the following order of execution:

Event	Order of execution
On opening the screen	- Screen_OnOpen() sub-routine from the screen script interface (VBScript language) - On opening from the screen logics interface (internal scripting language)
On closing the screen	- On closing from the screen logics interface (internal scripting language) - Screen_OnClose() sub-routine from the screen script interface (VBScript language)

20.4.5 MsgBox and InputBox Functions

The **MsgBox()** and **InputBox()** functions of the VBScript language enable displaying of pop-up messages during runtime. These functions are synchronous. If one of these functions is executed, the remaining instructions from the code are not executed before the pop-up messages launched by the function are closed.



The text displayed in these pop-up messages is not affected by the translation tool of "WinStudio", unless the text is explicitly configured using the \$Ext() function from the integrated scripting language.

20.4.6 VBScript Procedures

In VBScript, there are two kinds of procedures: Sub procedures and Function procedures.

Sub procedures Sub procedures are series of VBScript statements (enclosed by **Sub** and **End Sub** statements) that perform actions, however, do not return a value. Sub

Appendix: VBScript Language

procedures can contain arguments (constants, variables or expressions that are transferred by a calling procedure). If a Sub procedure has no arguments, its **Sub** statement has to include an empty set of parentheses ().

The following Sub procedure uses two intrinsic (integrated) VBScript functions, [MsgBox](#) and [InputBox](#) on page 744 to prompt a user input. Afterwards, the results of a calculation based on that information are displayed. The calculation is performed in a Function procedure created with VBScript. The Function procedure is displayed after the following discussion.

Sub procedure

```
Sub ConvertTemp()
    temp = InputBox("Please enter the temperature in degrees F.", 1)
    MsgBox "The temperature is " & Celsius(temp) & " degrees C."
End Sub
```

Function procedures

Function procedures are a series of VBScript statements enclosed by the **Function** and **End Function** statements. Function procedures are similar to Sub procedures, however, they can also return a value. Function procedures can contain arguments (constants, variables or expressions that are transferred to it by a calling procedure).

If a Function procedure has no arguments, its **Function** statement has to include an empty set of parentheses ().

A function returns a value by assigning a value to the function name in one or more statements of the procedure. The return value of a function is always a variant.

In the following example, the **Celsius** function calculates degrees Celsius from degrees Fahrenheit. If the function is called from the **ConvertTemp** Sub procedure, a variable containing the argument value is transferred to the function. The result of the calculation is returned to the calling procedure and displayed in a message box.

Function procedure

```
Sub ConvertTemp()
    temp = InputBox("Please enter the temperature in degrees F.", 1)
    MsgBox "The temperature is " & Celsius(temp) & " degrees C."
End Sub

Function Celsius(fDegrees)
    Celsius = (fDegrees - 32) * 5 / 9
End Function
```

Input and output of procedure data

Each data element is passed into the procedures using an argument. Arguments serve as placeholders for data to be passed into procedures. Arguments can be assigned any valid variable name. If a procedure is created using either the **Sub** statement or the **Function** statement, parentheses have to be included after the name of the procedure. Any arguments are placed inside these parentheses, separated by commas. In the following example, **fDegrees** is a placeholder for the value being passed into the **Celsius** function for conversion.

Input and output of procedure data

```
Function Celsius(fDegrees)
    Celsius = (fDegrees - 32) * 5 / 9
End Function
```

To receive data from a procedure, functions have to be used. Function procedures can return values while Sub procedures cannot.

Using Sub and Function procedures in code

Functions in codes have to be always used on the right side of variable assignments or in expressions. For example:

Appendix: VBScript Language

Using Function procedures in code

```
Temp = Celsius(fDegrees)
// or
MsgBox "The Celsius temperature is " & Celsius(fDegrees) & " degrees."
```

To call a Sub procedure from another procedure, enter the name of the procedure along with values for any required argument, each separated by a comma. The **Call** statement is not required. However, if it is used, all arguments have to be enclosed in parentheses.

The following example illustrates two calls for the **MyProc** procedure. One uses the **Call** statement in code, the other does not. Both have exactly the same effect.

Using Sub procedures in code

```
Call MyProc(firstarg, secondarg)

MyProc firstarg, secondarg
```

Please note that the parentheses are not specified in the call if the **Call** statement is not used.

20.4.7 Creating Constants

Constants are convenient names that replace numbers or Strings and never change. VBScript defines a number of intrinsic constants.

User-defined constants are created in VBScript using the Const statement.

By means of the Const statement, String or numeric constants with meaningful names can be created and get literal values assigned to them.

For example:

```
Const MyString = "This is my text."
```

```
Const MyAge = 49
```

Please note that the string is enclosed in quotation marks (" "). Quotation marks are the most obvious way to differentiate String values from numeric values. Date literals and time literals can be specified by enclosing them in number signs ("#").

For example:

```
Const EndDate = #6-1-97#
```

It is useful to apply a naming scheme to differentiate constants from variables. This helps to prevent reassignment of constant values while the script is running. For example, "vb" or "con" can be used as prefixes for constant names or constants may be named in capital letters. Clearly differentiating constants from variables helps to eliminate confusion when developing more complex scripts.

20.4.8 Declaring Variables

Variables are convenient placeholders that refer to a storage location on the computer where program information that may change during execution of the script can be stored. In VBScript, variables are always of the basic data type Variant.

Variables are explicitly declared in the script using the Dim statement, the Public statement, and the Private statement.

For example:

```
Dim DegreesFahrenheit
```

Multiple variables are declared by separating each variable name with a comma.

For example:

Dim Top, Bottom, Left, Right

A variable can also be declared implicitly by simply using its name in the script. In general, this is not good practice as the variable name could be misspelled in one or several places causing unexpected results executing the script. For this reason, the option Explicit statement is configured by default in the global procedures interface to require explicit declaration of all variables. Unless this statement is deleted, all variables have to be declared explicitly. Otherwise, VBScript generates errors during runtime indicating that the variable does not exist.

In an expression, the variable is to stand on the left and the value to be assigned to the variable on the right side.

For example:

MyVar = 100

20.4.9 Scope and Lifetime of Variables

The scope of a variable is determined by where it is declared. If a variable is declared within a procedure, only code within that procedure can access or change the value of the variable. It has local scope and is a procedure-level variable. If a variable is declared outside a procedure, it can be recognized by all procedures in the script. This is a script-level variable with script-level scope.

The lifetime of a variable depends on how long it exists. The lifetime of a script-level variable extends from the time it is declared until the time the script is finished running. At procedure level, a variable exists only as the procedure runs. The variable is destroyed on exiting the procedure. Local variables are ideal as temporary storage space when a procedure is running. Local variables with the same name can be declared to several different procedures as they are only recognized by the respective procedure in which they are declared.

20.4.10 Boolean Tags and Boolean Variables

By default, Boolean variables in VBScript are handled differently than [Boolean, tags page 113](#) in "WinStudio". The Boolean states FALSE and TRUE have the same meaning, however, the numeric values of TRUE are different as illustrated in the table below.

Boolean state	Numeric value in...	
	Project tag	VBScript variable
FALSE	0	0
TRUE	1	-1



In VBScript, "FALSE" and "TRUE" are reserved as [keywords on page 735](#).

This difference in how Booleans are handled can seriously affect runtime behavior if VBScript is used in the project. Logical and arithmetic operations – especially the [NOT operator on page 749](#) – could change tag values in unexpected ways.

Appendix: VBScript Language

Changing handling of Boolean tags in VBScript

If necessary, this behavior can be changed by adjusting the project runtime settings.

The handling of Boolean tags in VBScript can be changed by activating "VB Boolean" mode. To do so, change the following setting in the project file (**project_name.app**):

[Script]

VBBoolean=value

If **VBBoolean** is set to **0**, the project behaves as described above: All VBScript functions and operations read/write a value of **-1** for TRUE to Boolean tags. This is the default setting for projects created with "Rexroth WinStudio v7.0" or earlier that are updated to "Rexroth WinStudio v7.1" in order to maintain backward compatibility.

If **VBBoolean** is set to **1**, VBScript – as it is implemented in "WinStudio" – reads/writes a value of **1** for TRUE to Boolean tags. (This does not affect Integer or Real tags.) This is the default setting for projects created with "Rexroth WinStudio v7.1" or later.



Custom property on a [linked symbol on page 200](#) using the **#Label:@Pointer** syntax are to be defined with care.

For example:

#Label:@Pointer syntax

'The following statements are valid
If \$MyBoolean = 1 Then
End If

If \$MyBoolean = True Then
End If

If #Mne:@MyPointer = True Then
End If

'The following statement is invalid
If #Mne:@MyPointer = 1 Then
End If

20.4.11 Writing Real Values to Integer Tags

By default, a Real (i.e., floating point) value is cut off at the decimal point when it is written to an [Integer tag on page 113](#). This behavior is the same in both the integrated scripting language and in VBScript.

However, this behavior can be changed in VBScript by disabling the "TruncRealToInt" runtime setting. To do so, change the following line in the project file (**project_name.app**):

[Script]

TruncRealToInt=value

If **TruncRealToInt** is set to **"1"**, the project behaves as described above:

Real values are truncated at the decimal point without rounding. (For example: a value of "5.56" is written as "5" to an Integer tag.) This is the default setting for projects created with "Rexroth WinStudio v7.1" or earlier and then updated to "Rexroth WinStudio v7.2" in order to maintain backward compatibility.

If "TruncRealToInt" is set to **"0"**, VBScript functions and operations round Real values to the nearest integer.

(For example: a value of "5.56" is written as "6" to an Integer tag.) This is the default setting for projects created with "Rexroth WinStudio v7.2" or later.

Appendix: VBScript Language

This setting only affects the behavior of VBScript in "WinStudio". The behavior of the integrated scripting language is not affected.

20.4.12 Precedence of VBScript Operators

VBScript has a full range of operators, including arithmetic operators, comparison operators, concatenation operators and logical operators.

If several operations occur in an expression, each part is evaluated and resolved in a predetermined order called "operator precedence." Parentheses can be used to override the order of precedence and force some parts of an expression to be evaluated before others. Operations within parentheses are always performed first. Within the parentheses, however, standard operator precedence is maintained.

If expressions contain operators from more than one category, arithmetic operators are evaluated first, comparison operators are evaluated next and logical operators are evaluated last. All comparison operators have equal precedence, i.e., they are evaluated in a left-to-right order. Arithmetic and logical operators are evaluated in the following order of precedence.

Arithmetic	Comparison	Logical
Negation (-)	Equality (=)	Not
Exponentiation (^)	Inequality (<>)	And
Multiplication and division (*, /)	Less than (<)	Or
Integer division (\)	Greater than (>)	Xor
Modulus arithmetic (Mod)	Less than or equal to (<=)	Eqv
Addition and subtraction (+, -)	Greater than or equal to (>=)	Imp
String concatenation (&)	Is	&

If multiplication and division occur together in an expression, each operation is evaluated as it occurs from left to right. Likewise, if addition and subtraction occur together in an expression, each operation is evaluated in order of appearance from left to right.

The string concatenation (&) operator is not an arithmetic operator. In precedence, however, it is set after all arithmetic operators and before all comparison operators. The Is operator is an object reference comparison operator. It does not compare objects or their values. It checks only whether two object references refer to the same object.

20.4.13 Logical Operator NOT

The logical operator NOT behaves differently in VBScript than it does in the integrated scripting language.

NOT operator in VBScript

In VBScript, the NOT operator inverts the bits of a given numeric value producing its complement number according to the "two's complement" system of signed numbers that is used by computers. The following table illustrates the behavior of the NOT operator in VBScript for the syntax...

result = NOT expression

If "expression" is...	Then "result" is...
-3	2
-2	1

Appendix: VBScript Language

If "expression" is...	Then "result" is...
-1	0
0	-1
1	-2
2	-3
3	-4



If a numeric value not equal to 0 is to be written to a [Boolean tag on page 113](#), the tag is set to the value 1 by default.

For this reason, if the NOT operator of VBScript is applied to a Boolean tag with the value 1, the tag value does not change.

The operator returns a value of -2. The tag, however, cannot accept this value and is reset to 1. "WinStudio" can be configured to handle Boolean tags in VBScript similar to Boolean variables to ensure that the NOT operator in VBScript works as expected. For further information, please refer to [Boolean tags and Boolean variables on page 747](#).

NOT operator in integrated language

In contrast, the [NOT operator on page 484](#) in the integrated scripting language toggles the given numeric value as if it is a natural Boolean value. The following table illustrates the behavior of the NOT operator in the integrated scripting language for the syntax...

result = NOT expression

If "expression" is...	Then "result" is...
0	1
≠0	0

20.4.14 Using Conditional Statements

The flow of scripts can be controlled with conditional and looping statements. Using conditional statements, VBScript code can be written that makes decisions and repeats actions. The following conditional statements are available in VBScript:

- **If...Then...Else** statement
- **Select Case** statement

Making decisions using If...Then...Else

The **If...Then...Else** statement is used to evaluate whether a condition is **True** or **False** and, depending on the result, to specify one or more statements to run. Usually, the condition is an expression that uses a comparison operator to compare one value or variable with another. For information about comparison operators, refer to [Comparison operators](#).

If...Then...Else statements can be nested to as many levels as required.

Running statements if a condition is True

To run only one statement if a condition is True, the single-line syntax is used for the **If...Then...Else** statement. The following example illustrates the single-line syntax. Please note that this example does not include the **Else** keyword:

Single-line syntax for the If...Then...Else

```
Sub FixDate()  
    Dim myDate  
    myDate = #2/13/95#  
    If myDate < Now Then myDate = Now  
End Sub
```

To run more than one line of code, the multiple-line (or block) syntax is to be used. This syntax includes the **End If** statement, as illustrated in the following example:

Multiple-line (or block) syntax

```
Sub AlertUser(value)  
    If value = 0 Then  
        AlertLabel.ForeColor = vbRed  
        AlertLabel.Font.Bold = True  
        AlertLabel.Font.Italic = True  
    End If  
End Sub
```

Running certain statements if a condition is True and running others if a condition is False

A **If...Then...Else** statement can be used to define two blocks of executable statements: One block to run if the condition is **True** and the other block to run if the condition is **False**:

If...Then...Else statement to define two blocks...

```
Sub AlertUser(value)  
    If value = 0 Then  
        AlertLabel.ForeColor = vbRed  
        AlertLabel.Font.Bold = True  
        AlertLabel.Font.Italic = True  
    Else  
        AlertLabel.ForeColor = vbBlack  
        AlertLabel.Font.Bold = False  
        AlertLabel.Font.Italic = False  
    End If  
End Sub
```

Deciding between several alternatives

A variation on the **If...Then...Else** statement allows selecting between several alternatives. Adding **Elseif** clauses expands the functionality of the **If...Then...Else** statement. This way, the program flow can be controlled based on different possibilities. For example:

If...Then...Else statement with Elseif

```
Sub ReportValue(value)  
    If value = 0 Then  
        MsgBox value  
    ElseIf value = 1 Then  
        MsgBox value  
    ElseIf value = 2 then  
        MsgBox value  
    Else  
        MsgBox "Value out of range!"  
    End If  
End Sub
```

Elseif clauses can be added as required to provide alternative choices. However, extensive use of **Elseif** clauses often becomes cumbersome. A better way to choose between several alternatives is the **Select Case** statement.

Making decisions with Select Case

The "Select Case" structure provides an alternative to **If...Then...Elseif** for selectively executing one block of statements from multiple blocks of statements. A **Select Case** statement provides capability similar to the **If...Then...Else** statement. However, it makes the code more efficient and readable.

Appendix: VBScript Language

A **Select Case** structure works with a single test expression that is evaluated once at the top of the structure. The result of the expression is then compared to the values for each **Case** in the structure. If there is a match, the block of statements associated with that **Case** is executed, as in the following example:

Select Case structure

```
Select Case Document.Form1.CardType.Options(SelectedIndex).Text
  Case "MasterCard"
    DisplayMCLogo
    ValidateMCAccount
  Case "Visa"
    DisplayVisaLogo
    ValidateVisaAccount
  Case "American Express"
    DisplayAMEXCOLogo
    ValidateAMEXCOAccount
  Case Else
    DisplayUnknownImage
    PromptAgain
End Select
```

Please note that the **Select Case** structure evaluates an expression once at the top of the structure. In contrast, the **If...Then...Elseif** structure can evaluate a different expression for each "Elseif" statement. A **If...Then...Elseif** structure can only be replaced by a **Select Case** structure if each **Elseif** statement evaluates the same expression.

20.4.15 Looping with Code

Looping enables running a group of statements repeatedly. Some loops repeat statements until a condition is False. Other loops repeat statements until a condition is True. There are also loops that repeat statements a specific number of times.

The following looping statements are available in VBScript:

- **Do...Loop**: Loops while or until a condition is True
- **While...Wend**: Loops while a condition is True
- **For...Next**: Using a counter to run statements a specified number of times

Using Do Loops

"Do...Loop" statements can be used to run a block of statements an indefinite number of times. The statements are repeated either while a condition is True or until a condition becomes True.

Repeating statements while a condition is True

The While keyword can be used to check a condition in a **Do...Loop** statement. The condition can be checked before entering the loop (as shown in the following **ChkFirstWhile** example) or after the loop has run at least once (as shown in the **ChkLastWhile** example). In the **ChkFirstWhile** procedure, if **myNum** is set to 9 instead of 20, the statements inside the loop are never executed. In the **ChkLastWhile** procedure, the statements inside the loop run only once as the condition is already False.

Repeating statements while a condition is True

```
Sub ChkFirstWhile()
  Dim counter, myNum
  counter = 0
  myNum = 20
  Do While myNum > 10
    myNum = myNum - 1
    counter = counter + 1
  Loop
  MsgBox "The loop made " & counter & " repetitions."
End Sub
```



```
Sub ChkLastWhile()  
    Dim counter, myNum  
    counter = 0  
    myNum = 9  
    Do  
        myNum = myNum - 1  
        counter = counter + 1  
    Loop While myNum > 10  
    MsgBox "The loop made " & counter & " repetitions."  
End Sub
```

Repeating a statement until a condition becomes True

There are two ways to use the **Until** keyword to check a condition in a **Do...Loop** statement. The condition can be checked before entering the loop (as shown in the following **ChkFirstUntil** example) or after the loop has run at least once (as shown in the **ChkLastUntil** example). As long as the condition is **False**, the looping occurs.

Repeating a statement until a condition becomes True

```
Sub ChkFirstUntil()  
    Dim counter, myNum  
    counter = 0  
    myNum = 20  
    Do Until myNum = 10  
        myNum = myNum - 1  
        counter = counter + 1  
    Loop  
    MsgBox "The loop made " & counter & " repetitions."  
End Sub
```

```
Sub ChkLastUntil()  
    Dim counter, myNum  
    counter = 0  
    myNum = 1  
    Do  
        myNum = myNum + 1  
        counter = counter + 1  
    Loop Until myNum = 10  
    MsgBox "The loop made " & counter & " repetitions."  
End Sub
```

Exiting a Do...Loop statement from inside the loop

A **Do...Loop** can be exited by using the **Exit Do** statement. As loops are typically only exited in certain situations, for example, to avoid an endless loop, the **Exit Do** statement is to be used in the **True** statement block of an **If...Then...Else** statement. If the condition is **False**, the loop runs as usual.

In the following example, "myNum" is assigned a value that creates an endless loop. The **If...Then...Else** statement checks for this condition, preventing the endless repetition.

Exiting a Do...Loop statement from inside the loop

```
Sub ExitExample()  
    Dim counter, myNum  
    counter = 0  
    myNum = 9  
    Do Until myNum = 10  
        myNum = myNum - 1  
        counter = counter + 1  
        If myNum < 10 Then Exit Do  
    Loop  
    MsgBox "The loop made " & counter & " repetitions."  
End Sub
```

Appendix: VBScript Language

Using While...Wend

The **While...Wend** statement is provided in VBScript for experienced users. However, due to the lack of flexibility of **While...Wend**, it is recommended to use **Do...Loop** instead.

Using For...Next

For...Next statements can be used to run a block of statements a specific number of times. For loops, a counter variable is to be used whose value increases or decreases with each repetition of the loop.

The following example causes a procedure called **MyProc** to execute 50 times. The For statement specifies the counter variable **x** and its start and end values. The "Next" statement increments the counter variable by 1.

Using For...Next (1)

```
Sub DoMyProc50Times()
    Dim x
    For x = 1 To 50
        MyProc
    Next
End Sub
```

Using the **Step** keyword, the counter variable can be increased or decreased by the specified value. In the following example, the counter variable **j** is increased by 2 each time the loop repeats. If the loop is finished, the total is the sum of 2, 4, 6, 8, and 10.

Using For...Next (2)

```
Sub TwosTotal()
    Dim j, total
    For j = 2 To 10 Step 2
        total = total + j
    Next
    MsgBox "The total is " & total
End Sub
```

To decrease the counter variable, a negative **Step** value is to be used. An end value that is less than the start value is to be specified. In the following example, the counter variable **myNum** is decreased by 2 each time the loop repeats. If the loop is finished, the total is the sum of 16, 14, 12, 10, 8, 6, 4, and 2.

Using For...Next (3)

```
Sub NewTotal()
    Dim myNum, total
    For myNum = 16 To 2 Step -2
        total = total + myNum
    Next
    MsgBox "The total is " & total
End Sub
```

The **For...Next** statement can be exited at any time before the counter reaches its end value by using the **Exit For** statement. As loops are typically only exited in certain situations, for example, in case of an error, the **Exit For** statement is to be used in the **True** statement block of an **If...Then...Else** statement. If the condition is **False**, the loop runs as usual.

20.4.16 Support for ActiveX Control Elements

Using the VBScript interfaces for the Graphic module (graphic script, screen script, command animation and ActiveX events), this syntax can be applied to access properties and methods directly from any ActiveX control object in the screen, in which the object is configured.

Appendix: VBScript Language

"WinStudio" assigns a unique name to the object on the screen. The Name property can be used (in the **Object Properties** dialog) to modify this name.

After inserting an ActiveX control object on the screen, the properties and methods of this object can be accessed from any VBScript interface associated with this screen. Use the syntax **Object_Name.Properties_or_Method_Name**.

Examples:

```
//Access the value of the property Day from the CalendarControl1 ActiveX object  
CalendarControl1.Day
```

```
//Triggers the method AboutBox from the CalendarControl1 ActiveX object  
CalendarControl1.AboutBox
```

20.4.17 Windows Embedded Support

CEView also supports VBScript. The hardware manufacturer of the Microsoft Windows Embedded device has to enable support of VBScript on the device to enable CEView to execute the scripts configured in the VBScript language on the device.

The **MsgBox()** and **InputBox()** functions can be specifically enabled/disabled by the hardware manufacturer when the image for the Windows Embedded device is created.

In case of doubt whether the image loaded on the device supports VBScript, please contact the hardware manufacturer.

21 Appendix: Driver Error Codes

The **Read Status** and **Write Status** fields return error codes to the tags specified for these fields. These error codes can be default codes or customized for each driver. For a description of customized error codes, please refer to the documentation for the driver.

The following values are default error codes, which are always negative values:

Value	Description
0	OK
-1	Invalid serial port
-2	Invalid baud rate
-3	Invalid number of bits
-4	Invalid number of stop bits
-5	Invalid parity
-6	Invalid irq
-7	Serial port already in use
-8	Invalid buffer size
-9	Memory not enough
-10	Tx buffer empty
-11	Tx buffer full
-12	Rx buffer empty
-13	Rx buffer full
-14	Timeout waiting CTS
-15	Timeout waiting start a message
-16	Timeout waiting a message to finish
-17	Timeout between rx char
-18	Timeout between tx char
-19	No carrier detected
-20	No DSR detected
-21	Could not find a 8250 in address
-22	Tx line is busy
-23	User abort
-24	Function not supported
-25	Overrun
-26	Parity
-27	Overrun and parity
-28	Framing
-29	Framing and overrun

Appendix: Driver Error Codes

Value	Description
-30	Framing and parity
-31	Framing, overrun and parity
-32	Timeout waiting a tx message to finish

The **UNICOMM.MSG** file in the [...]WinStudio\Bin directory also contains descriptions of the default error codes.

The **DriverName.MSG** file in the [...]WinStudio\Drv directory contains descriptions of the customized error codes for each driver.

22 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

Symbols

.NET controls..... 214

A

About this documentation..... 19
Validity of this documentation..... 19

Abs..... 485

ACos..... 525

Active objects

Button..... 176

Checkbox..... 181

Combo box..... 188

List box..... 192

Pushbutton..... 178

Radio button..... 184

Smart message..... 195

Text..... 175

ActiveX and .NET control functions..... 683

ActiveX control..... 208

ActiveX events..... 722

Add a link to an external image file..... 219

Adjust the position of a screen object..... 239

Adjust the size of a screen object..... 239

Advanced settings for alarm worksheet..... 252

Alarm history

Format..... 255

Alarm worksheet..... 247

Advanced settings..... 252

Email settings..... 250

Alarm worksheet body..... 253

Alarm worksheet header..... 248

Alarm/event control object..... 262

Alarms, events and trends..... 247

Align, center and distribute..... 236

Animations..... 164

Bargraph..... 224

Color..... 227

Command..... 220

Hyperlink..... 224

Position..... 229

Resize..... 231

Rotation..... 232

Text data link..... 225

Apache for Windows..... 410

AppActivate..... 547

Appendix

Built-in scripting language..... 471

Driver error codes..... 757

VBScript..... 715

Appendix_A..... 298

Appendix_B..... 301

ApplsRunning..... 548

Apply animations to screen objects..... 220

AppPostMessage..... 549

AppSendKeys..... 550

Arithmetic functions..... 484

Arithmetic operators..... 484

Array tags..... 114

Asc2Str..... 504

ASin..... 525

Asynchronous

versus synchronous..... 574

Auto declaration

WinStudio tags..... 122

AutoFormat..... 597

Automatic translation..... 381

Automatically run a project..... 428

Avg..... 495

B

Background color of a screen..... 160

Background picture..... 160

Background Task

Execute..... 34

Switch..... 34

Background tasks..... 315

Bargraph animation..... 224

Batch file

Calling an external file..... 576

Block or unblock users..... 379

BlockUser..... 535

Boolean tags..... 747

Boolean variables..... 747

Button object..... 176

C

CEWebServer..... 408

Changing the target system of an existing project..... 91

CharToValue..... 505

CharToValueW..... 505

Checkbox

Operating modes..... 186

Checkbox object..... 181

CheckESign..... 535

Classes

Create..... 123

Tag..... 117

Classes folder..... 108

ClassMembersToStrVector..... 506

CleanReadQueue..... 550

Client/server architecture..... 399

ClockGetDate..... 520

ClockGetDayOfWeek..... 521

ClockGetTime..... 522

Close..... 528

Closed polygon object..... 169

CloseSplashWindow..... 550

CnfEmail..... 658

CnfFTP..... 687

Index

Color animation.....	227	DBCursorColumnCount.....	694
Color interface.....	241	DBCursorColumnInfo.....	694
Combo box object.....	188	DBCursorCurrentRow.....	695
Comm tab.....	58	DBCursorGetValue.....	696
Command animation.....	220, 721	DBCursorMoveTo.....	697
Communication tab.....	74	DBCursorNext.....	698
Communication with a remote device.....	333	DBCursorOpen.....	699
Communication with other devices.....	333	DBCursorOpenSQL.....	700
Communication with remote device		DBCursorPrevious.....	702
Default driver sheets.....	339	DBCursorRowCount.....	703
Conditional statements.....	750	DBDelete.....	703
Configuring a DDE client connection to a		DBExecute.....	704
DDE server.....	352	DBInsert.....	706
Configuring a TCP/IP client connection to		DBSelect.....	707
another project.....	351	DBUpdate.....	709
Configuring an OPC client connection to an		DbVersion.....	611
OPC server.....	342	DeleteOlderFiles.....	577
Configuring an OPC UA client connection to		Deleting a tag	
an OPC UA server.....	345	from the project database.....	135
Constants.....	726	Development environment	
Creating.....	746	Winstudio.....	49
Control layer		Development station	
ISSymbol.....	397	Configuration.....	426
Conventions used in this documentation.....	21	Dial-up functions.....	664
Converting trend history files		DialError.....	664
from binary to text.....	272	DialGetClientIP.....	670
from text to binary.....	273	DialGetServerIP.....	672
Cos.....	526	DialStatus.....	673
Cot.....	526	DialUp.....	675
Create and configure groups.....	370	DialUpToCE.....	676
Create and configure users.....	374	DirCreate.....	578
Create batch history.....	273	DirDelete.....	579
Create tags		DirLength.....	579
Datasheet.....	121	DirRename.....	580
CreateUser.....	536	DisableMath.....	551
Creating a new project.....	89	Div.....	485
Creating tags		Documentation structure.....	19
"On-the-fly".....	122	Driver error codes.....	757
Modification.....	122		
Cross reference list		E	
Winstudio.....	87	Edit group	
D		Arrange objects.....	166
Data server		Deactivate dragging into a screen.....	166
Configuration.....	403	Grid settings.....	167
Data type		Object properties.....	164
WinStudio.....	113	Replace project tags.....	136, 164
Database debugging.....	443	Undo.....	166
Database interface.....	429	Edit screen objects.....	164
Database spy.....	84	Ellipse object.....	172
Database Spy.....	387	Email	
Remote.....	388	Configuring.....	658
Database/ERP functions.....	692	Email functions.....	658
Database/ERP worksheet.....	328	Email settings for alarm worksheet.....	250
Date and time functions.....	520	EnableMath.....	551
DateTime2Clock.....	522	EndTask.....	551
DBCursorClose.....	693	Error codes	
		Driver.....	757

Errors.....	735
Event log	
Format.....	260
Event Logger.....	258
Event logger functions.....	686
Events and event logs.....	258
Execute.....	60
ExecuteAlarmAck.....	637
Execution modes.....	44
Exercise	
View the project on the web.....	422
ExitWindows.....	552
Exp.....	498
ExportSecuritySystem.....	537
Ext.....	607

F

False.....	501
File functions.....	577
FileCopy.....	581
FileDelete.....	581
FileLength.....	582
FileRename.....	582
FileWrite.....	583
Fill color.....	240
FindAllDevices.....	678
FindFile.....	584
FindModem.....	678
FindPath.....	585
Folder	
Classes.....	108
Project tags.....	106
Shared Database.....	109
System Tags.....	110
Fonts tool.....	244
For ... Next.....	640
ForceTagChange.....	638
Format.....	486
Format tab.....	233
Frequently asked questions.....	462
FTP functions.....	687
FTPGet.....	688
FTPPut.....	690
FTPStatus.....	691
Functional descriptions	
How to read.....	482
Functions.....	734

G

GetAppHorizontalResolution.....	611
GetAppPath.....	612
GetBit.....	489
GetClock.....	523
GetComputerIP.....	612
GetComputerName.....	613
GetCursorX.....	613
GetCursorY.....	613

GetDisplayHorizontalResolution.....	614
GetDisplayVerticalResolution.....	614
GetFileAttributes.....	585
GetFileTime.....	586
GetFreeMemoryCE.....	614
GetHardKeyModel.....	615
GetHardKeySN.....	615
GetHstInfo.....	586
GetIPAll.....	616
GetLine.....	587
GetMemoryCE.....	616
GetNetMACID.....	617
GetOS.....	618
GetPrivateProfileString.....	618
GetProductPath.....	619
GetRegValue.....	619
GetRegValueType.....	620
GetScrInfo.....	598
GetSecuritySystemStatus.....	538
GetServerHostName.....	621
GetStatusSendEmailExt.....	661
GetTagValue.....	638
GetTickCount.....	621
GetUserFullName.....	539
GetUserNames.....	540
GetUserPwdAging.....	541
GetUserState.....	541
Global procedures.....	716
Global Replace tool.....	135
Global tab.....	56
Graphic functions.....	597
Graphics	
Active objects.....	174
Animations.....	220
Shapes.....	167
Graphics tab.....	55, 156
Grid object.....	305
Advanced dialog.....	311
Columns dialog.....	307
Data dialog.....	309
Group and ungroup objects.....	235

H

HangUp.....	679
Hour2Clock.....	523
HST2TXT.....	590
HST2TXTIsRunning.....	591
Hyperlink animation.....	224

I

If.....	501
IIS.....	408
Import from CSV databases.....	146
Import from ODBC databases.....	147
Import from other project databases.....	143
Import Wizard.....	140
Importing from OPC server databases.....	145

Index

ImportSecuritySystem.....	542
Indirect tags (pointers).....	119
InfoAppAlrDir.....	622
InfoAppHSTDir.....	622
InfoDiskFree.....	622
InfoResources.....	623
Information tab.....	62
Installation.....	37
System requirements.....	37
Installing the Software.....	38
IntelliSense	
VBScript editor.....	740
IsActiveXReg.....	623
IsAppChangedOnServer.....	624
IsScreenOpen.....	553
ISSymbol control layer.....	397
IsTaskRunning.....	553
IsViewerInFocus.....	554

K

Keyboard.....	73
Virtual.....	73
KeyPad.....	554
Keywords.....	735

L

Language reference.....	725
Constants.....	726
Errors.....	735
Functions.....	734
Keywords.....	735
Methods.....	733
Objects and collections.....	730
Properties.....	730
Statements.....	731
System tags.....	738
Language Reference	
Operators.....	725
Libraries	
.NET controls.....	214
ActiveX control.....	208
Symbols.....	199, 200

License

invalid.....	47
License protection types.....	43
License settings.....	43
Licensing.....	43
Lifetime of variables.....	747
Line color.....	241
Line object.....	167
Link the database through a remote DB pro- vider.....	431
List box object.....	192
List of available functions.....	471
List of system tags.....	738
Log.....	499
Log message functions.....	483

Log on/off.....	378
Log10.....	499
Logarithmic functions.....	498
Logical functions.....	500
Logical operator NOT.....	749
Logical operators.....	500
LogOff.....	556
LogOn.....	556
LogWin	
Remote.....	389
LogWin output window.....	85
LookupContains.....	592
LookupGet.....	593
LookupLoad.....	593
Loop functions.....	640
Looping with code.....	752

M

Math.....	557
Math worksheet.....	324
Max.....	496
Methods.....	733
Microsoft Access databases	
Use.....	451
Microsoft Excel	
Use.....	456
Microsoft SQL Server	
Use.....	448
Min.....	497
Mirror horizontally.....	238
Mirror vertically.....	238
Mobile access.....	407
Mod.....	490
Module activity functions.....	545
MsgBox and InputBox functions.....	744
Multimedia functions.....	608
MySQL	
Use.....	458

N

NCopy.....	507
Next.....	640
NoInputTime.....	625
NTWebServer.....	408
Num.....	508

O

Object Finder tool.....	139
Object properties	
.NET controls.....	214
ActiveX control.....	208
Animation.....	232
Bargraph.....	224
Button.....	176
Checkbox.....	181
Closed polygon.....	169
Colors.....	227

Index

Combo box.....	188
Command.....	220
Ellipse.....	172
Hyperlink.....	224
Line.....	167
List box.....	192
Open polygon.....	168
Position.....	229
Pushbutton.....	178
Radio button.....	184
Rectangle.....	170
Rounded rectangle.....	171
Size.....	231
Smart message.....	195
Text.....	175
Text data link.....	225
Object properties dialog.....	164
Objects and collections.....	730
ODBC databases	
Use.....	447
ODBC functions.....	640
ODBC worksheet.....	321
ODBCBeginTrans.....	642
ODBCBindCol.....	642
ODBCCanAppend.....	644
ODBCCanUpdate.....	644
ODBCClose.....	645
ODBCCommitTrans.....	645
ODBCDelete.....	646
ODBCExecuteSQL.....	646
ODBCInsert.....	647
ODBCIsBOF.....	648
ODBCIsDeleted.....	648
ODBCIsEOF.....	649
ODBCIsFieldNULL.....	649
ODBCIsFieldNullable.....	650
ODBCMove.....	650
ODBCMoveFirst.....	651
ODBCMoveLast.....	651
ODBCMoveNext.....	652
ODBCMovePrev.....	652
ODBCOpen.....	653
ODBCQuery.....	654
ODBCRollBack.....	654
ODBCSetFieldNULL.....	655
ODBCSetFilter.....	656
ODBCSetSort.....	656
ODBCUnbindCol.....	657
ODBCUpdate.....	657
Open.....	529
Open polygon object.....	168
OpenPrevious.....	531
Operators.....	725
Arithmetic.....	484
Logical.....	500
Option tab	
Configure database for all task histories	64, 442
Options tab.....	62

Alarm history.....	63, 254
Database configuration.....	65, 438
Events.....	259
ORACLE databases	
Use.....	449
Output (LogWin).....	385

P

Password protection	
Screens, symbols and worksheets.....	379
PDFCreate.....	594
PhoneDialUp.....	680
PhoneDisableListen.....	680
PhoneEnableListen.....	681
PhoneHangUp.....	682
PhoneStatus.....	682
Pi.....	527
Play.....	609
Pointers.....	119
Port Usage.....	421
Position animation.....	229
PostKey.....	557
Pow.....	490
Preferences tab.....	76
Print.....	595
PrintSetup.....	599
PrintWindow.....	599
Product features.....	26
Product overview.....	25
Product type.....	91
Product versions.....	45
ProductVersion.....	625
Project execution	
Windows services.....	95
Project Explorer.....	55
Comm tab.....	58
Global tab.....	56
Graphics tab.....	55, 156
Tasks tab.....	57
Project language	
Setting at program start.....	383
Setting at Runtime.....	383
Project security.....	357
Project settings	
Additional configuration.....	62
Communication tab.....	74
E-Mail.....	92
FTP configuration.....	94
Information tab.....	62
Options tab.....	62
Preferences tab.....	76
Viewer tab.....	69
Web tab.....	78
Project tags datasheet	
Extend.....	107
Project tags folder.....	106
Properties.....	730

Index

Pushbutton object..... 178

R

Radio button
 Operating modes..... 186
 Radio button object..... 184
 Rand..... 498
 RDNFileN..... 595
 Real values to Integer tags..... 748
 Recipe..... 558
 Recipe worksheet..... 319
 Rectangle object..... 170
 RegSaveCE..... 626
 ReloadAppFromServer..... 626
 Remote Database Spy..... 388
 Remote LogWin..... 389
 Remote management..... 61
 Remove unused tags..... 137
 RemoveUser..... 543
 Replace project tags in a document or
 screen object..... 136, 164
 Report..... 558
 Report worksheet on page..... 320
 ResetBit..... 491
 ResetDecimalPointsTable..... 601
 Resize animation..... 231
 RGBColor..... 601
 RGBComponent..... 602
 Rotating..... 238
 Rotation animation..... 232
 Round..... 492
 Rounded rectangle object..... 171
 Run a project..... 428
 RunGlobalProcedureAsync..... 560
 RunGlobalProcedureAsyncGetStatus..... 561
 RunGlobalProcedureOnFalse..... 563
 RunGlobalProcedureOnTrigger..... 565
 RunGlobalProcedureOnTrue..... 565
 RunVBScript..... 566

S

Safety alert symbol..... 21
 Safety instructions..... 20, 21
 Save own project symbols..... 200
 SaveAlarmFile..... 627
 SaveScreenShot..... 602
 Scheduler worksheet..... 326
 Scope of variables..... 747
 Screen attributes dialog..... 156
 Screen events..... 744
 Screen functions..... 528
 Screen groups folder..... 161
 Screen objects..... 164
 Active objects..... 174
 Animations..... 220
 Disable translation..... 384
 Shapes..... 167

Screen script..... 720
 Screen/worksheet Editor..... 84
 Screens and graphics..... 155
 Screens, symbols and worksheets
 Password-protected..... 379
 Script worksheet..... 324
 Startup..... 325
 SecureViewerReload..... 567
 Security..... 357
 Security access levels..... 358
 Security for web-based applications
 Implementation..... 417
 Security functions..... 534
 Security modes..... 357
 Server settings..... 362
 Security rights
 Saving..... 365
 Security System
 Configuration Wizard..... 359
 Security system configuration
 Data backup..... 377
 Security system dialog..... 375
 SendEmail..... 661
 SendEmailExt..... 662
 SendEvent..... 686
 SendKeyObject..... 568
 SetAppAlarmPath..... 628
 SetAppHSTPath..... 628
 SetAppPath..... 570
 SetBit..... 492
 SetDateFormat..... 629
 SetDecimalPoints..... 604
 SetDisplayUnit..... 605
 SetKeyboardLanguage..... 629
 SetLanguage..... 607
 SetPassword..... 544
 SetRegValue..... 630
 SetSystemDate..... 524
 SetSystemTime..... 524
 SetTagDisplayUnit..... 606
 SetTagValue..... 639
 SetTranslationFile..... 608
 SetViewerInFocus..... 570
 SetViewerPos..... 571
 SetWebConfig..... 631
 Shapes group..... 167
 Add graphics to a screen..... 173
 Closed polygon object..... 169
 Ellipse object..... 172
 Line object..... 167
 Open polygon object..... 168
 Rectangle object..... 170
 Rounded rectangle object..... 171
 Shared Database folder..... 109
 Sharing tags
 IndraLogic devices..... 149
 ShowInplaceInput..... 532
 ShowMessageBox..... 533

ShutDown.....	572	System Tags folder.....	110
Signal words.....	21		
Sin.....	527	T	
SMA.....	407	Tag data type	
Smart message object.....	195	Choice.....	113
SNMPGet.....	633	Tag integration.....	148
SNMPSet.....	634	Tag name syntax.....	111
SQL relational databases.....	430	Tag properties	
SQL Server CE		List.....	129
Use.....	453	Tag properties and parameters.....	124
SQL Server on a Windows Embedded device..	454	Tag Properties tool.....	140
Sqrt.....	493	Tag properties: Alarms	
Starting the Engineering software.....	42	Use.....	126
StartTask.....	572	Tag properties: History	
Statements.....	731	Use.....	129
Statistical functions.....	495	Tag scope	
Status bar		Choice.....	120
Winstudio.....	87	Tag type	
Status codes for external database operations.	330	Choice.....	111
Stop.....	60	Tags and the project database.....	105
Str.....	508	Tags database functions.....	636
Str2Asc.....	509	Tags database reset.....	138
StrCompare.....	509	Tags in the project	
StrCompareNoCase.....	510	Use.....	134
StrFromInt.....	510	Tan.....	527
StrFromReal.....	511	Target platforms and product types.....	90
StrFromTime.....	512	Target system	
StrGetElement.....	513	Configuration.....	425
String functions.....	503	Tasks tab.....	57
StrLeft.....	513	Text data link animation.....	225
StrLen.....	514	Text object.....	175
StrLower.....	514	Thin Client	
StrRChr.....	514	Configuration.....	413
StrRight.....	515	Thin Client program.....	393
StrSetElement.....	515	Thin Clients.....	391
StrStr.....	516	Tips & Tricks.....	740
StrStrPos.....	516	To back, by one level.....	235
StrTrim.....	517	To front, by one level.....	235
StrTrimAll.....	518	To the foreground and to the background.....	234
StrUpper.....	518	Toggle.....	502
Studio Database Gateway.....	432	Toolbar	
Support		WinStudio alignment functions.....	51
See service hotline.....	759	WinStudio debug functions.....	53
Swap16.....	493	WinStudio dynamic functions.....	54
Swap32.....	494	WinStudio execution functions.....	50
Sybase		WinStudio object properties.....	52
Use.....	455	WinStudio table functions.....	53
Symbols.....	21, 199	WinStudio tag properties.....	52
SyncAlarm.....	710	WinStudio, customize.....	49
SyncAlarmStatus.....	711	Tools	
SyncEvent.....	711	Winstudio.....	82
SyncEventStatus.....	712	Trace.....	483
Synchronous		Translation functions.....	606
versus asynchronous.....	574	Translation Table	
SyncTrendStatus.....	714	Add a language.....	381
System information functions.....	609	Trend control	
System requirements.....	37	Advanced.....	295
System tags.....	738		

Index

Axes.....	287
Legend.....	294
SPC.....	286
Toolbar.....	291
Trend database.....	276
Trend display.....	277
Trend object	
Data source.....	292
Object properties.....	281
Options.....	285
Pen style.....	283
Points.....	282
Use a database as data source.....	301
Use a text file as data source.....	298
Trend object runtime interface.....	278
Trend worksheet.....	269
Trigonometric functions.....	524
Troubleshooting.....	461
True.....	503
Trunc.....	494

U

UnblockUser.....	545
Use the LogWin module.....	387

V

ValueToChar.....	519
ValueWToChar.....	519
Variables	
Declaring.....	746
VBScript.....	715
Calling an external file.....	576
VBScript editor IntelliSense.....	740
VBScript interfaces in the Software.....	715
VBScript operators	
Precedence of	749
VBScript procedures.....	744
Verify project.....	59
Viewer tab.....	69
ViewerPostMessage.....	573
Virtual keyboard.....	73

W

Wait.....	574
Web application	
Page server configuration.....	407
Web pages folder.....	162
Web server	
Apache for Windows.....	410
Installation and configuration.....	407
Web server as host for project pages	
Configuration.....	407
Web servers	
CEWebServer.....	408
Microsoft IIS.....	408
Web Servers	
NTWebServer.....	408

Web tab.....	78
Web Tunneling Gateway.....	407
Installation.....	410
WebGetFile.....	596
Windows	
Processes.....	660
Windows command	
Execution.....	574
Windows Embedded support.....	755
WinExec.....	574
WinExecIsRunning.....	576
WinStudio folder.....	156
WinStudio Libraries.....	199
WinStudio toolbar	
Alignment functions.....	233
WritePrivateProfileString.....	636

X

XGet.....	684
XRun.....	684
XSet.....	685

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



R911336882

DOK-IWORKS-WINSTUD*V13-AP01-EN-P