

RSLogix
Automation Interface
Reference Manual

June-2002



Contacting Technical Support Telephone—1-440-646-5800
Rockwell Software Technical Support Fax—1-440-646-5801
World Wide Web—www.software.rockwell.com

Copyright Notice © 2001, 2002 Rockwell Software Inc., a Rockwell Automation company. All rights reserved
Printed in the United States of America
Portions copyrighted by Allen-Bradley Company, LLC, a Rockwell Automation company.
This manual and any accompanying Rockwell Software products are copyrighted by Rockwell Software Inc. Any reproduction and/or distribution without prior written consent from Rockwell Software Inc. is strictly prohibited. Please refer to the license agreement for details.

Trademark Notices The Rockwell Software logo, RSLogix 5, RSLogix 500, RSVIEW32, RSVIEW, and SoftLogix 5 are trademarks of Rockwell Software Inc., a Rockwell Automation company.
DH+, MicroLogix, PLC, PLC-2, PLC-5, PLC-5/250, SLC, and SLC 500 are trademarks of the Allen-Bradley Company, LLC, a Rockwell Automation company.
Microsoft, and Visual Basic are registered trademarks of the Microsoft Corporation.
ControlNet is a trademark of ControlNet International.
All other trademarks are the property of their respective holders and are hereby acknowledged.

Warranty This Rockwell Software product is warranted in accord with the product license. The product's performance will be affected by system configuration, the application being performed, operator control and other related factors.
The product's implementation may vary among users.
This manual is as up-to-date as possible at the time of printing; however, the accompanying software may have changed since that time. Rockwell Software reserves the right to change any information contained in this manual or the software at anytime without prior notice.
The instructions in this manual do not claim to cover all the details or variations in the equipment, procedure, or process described, nor to provide directions for meeting every possible contingency during installation, operation, or maintenance.

Contents

Chapter 1

Introduction to the automation interface 1

What is VBA and what does it do?	1
Advantages	1
Uses	2
Finding your way around this book	3
Automating the ladder logic editor.....	3
Automating the documentation database editor	4
Supplemental information	5
Example files.....	5
How to access VBA in RSLogix 5 and RSLogix 500	6
Create your VBA code	6
Some quick programming tips	6

Chapter 2

Application object 9

Properties	10
Methods.....	13
Events	18
Summary example	22
Form.....	22
Code	23

Chapter 3

LogixProject object 25

Properties	26
Methods.....	28
Events	36
Summary example	41
Form.....	41

Code	41
------------	----

Chapter 4

Processor object45

Properties	46
Methods	49
Events.....	51
Summary example.....	51
Form.....	52
Code	53

Chapter 5

ProgramFiles collection57

Properties	57
Methods	58
Events.....	60
Summary example.....	60
Form.....	61

Chapter 6

ProgramFile object.....65

Properties	66
Methods	68
Events.....	68
Summary example.....	68
Form.....	69
Code	70

Chapter 7

DataFiles collection75

Properties	75
Methods	76
Events.....	79
Summary Example	79
Form.....	80

Code	81
------------	----

Chapter 8

DataFile object 85

Properties	86
Methods	88
Events	88
Summary Example	89
Form	90
Code	90

Chapter 9

LadderFile object..... 95

Properties	96
Methods	98
Events	101
Summary example	101
Form	102
Code	103

Chapter 10

Rung object..... 109

Properties	110
Methods	112
Events	112
Summary example	112
Form	113
Code	114

Chapter 11

RevisionNotes object 119

Properties	120
Methods	120
Events	121
Summary example	121

Form.....	122
Code.....	122

Chapter 12

ReportOptions object127

Properties.....	128
Methods.....	132
Events.....	132
Summary example.....	133
Form.....	133
Code.....	133

Chapter 13

AddrSymRecords collection137

Properties.....	138
Methods.....	138
Events.....	144

Chapter 14

AddrSymRecord object145

Properties.....	146
Methods.....	147
Events.....	152

Chapter 15

RungCmntpageTitleRecords collection153

Properties.....	154
Methods.....	154
Events.....	164

Chapter 16

RungCmntpageTitleRecord object165

Properties.....	166
Methods.....	167
Events.....	169

PasswordPrivilegeConfig object	171
Properties	172
Methods	172
Events	189

Appendix A

Object model diagrams	191
Introduction	191
RSLogix 5 object model summary	192
RSLogix 500 object model summary	195
RSLogix 500 object model summary, database utilities	197

Appendix B

Type definitions and constants	199
RSLogix 5 and RSLogix 500 type definitions and constants	199
lgxDataFileTypeConstants (RSLogix 5)	200
lgxDataFileTypeConstants (RSLogix 500)	201
lgxKeyPositionConstants (RSLogix 5 and 500)	202
lgxOnlineAction (RSLogix 5 and 500)	202
lgxProcessorTypeConstants (RSLogix 5)	203
lgxProcessorTypeConstants (RSLogix 500)	204
lgxProcOnlineState (RSLogix 5)	205
lgxProcOnlineState (RSLogix 500)	205
lgxProgramFileTypeConstants (RSLogix 5)	206
lgxProgramFileTypeConstants (RSLogix 500)	206
lgxRungZoneTypes (RSLogix 5 and 500)	206
lgxSaveAction (RSLogix 5 and 500)	206
lgxUpDownAction (RSLogix 5 and 500)	207
lgxWindowStateConstants (RSLogix 5 and 500)	207
lgxImportDBTypes (RSLogix 5 and 500)	207
lgxBinary (RSLogix 5)	207
lgxChannel (RSLogix 5)	207
lgxPrivilege (RSLogix 5)	208

lgxPrivilegeType (RSLogix 5)	208
lgxErrorTypes (RSLogix 5 and 500)	209

Appendix C

Handling errors.....211

Appendix D

General differences in the RSLogix 5 and 500 automation interfaces215

PasswordPrivilegeConfig	216
DataFile object	216
ProgramFile object.....	216
ReportOptions object.....	216
LogixProject object.....	216
Processor object	217
Ladder object.....	217

Index.....219

Introduction to the automation interface

What is VBA and what does it do?

Visual Basic® for Applications (VBA) is the edition of Visual Basic designed specifically to provide rich development capabilities in an off-the-shelf application. Microsoft® licenses VBA to application vendors such as Rockwell Software, who integrate it into their products. This makes the familiar Visual Basic development environment readily available for users to adopt, rapidly extending their host application and integrating it with other VBA enabled applications. In this sense, VBA is a “glue” or bridge between Component Object Model (COM)-enabled software packages that allows them to efficiently inter-operate with each other.

Advantages

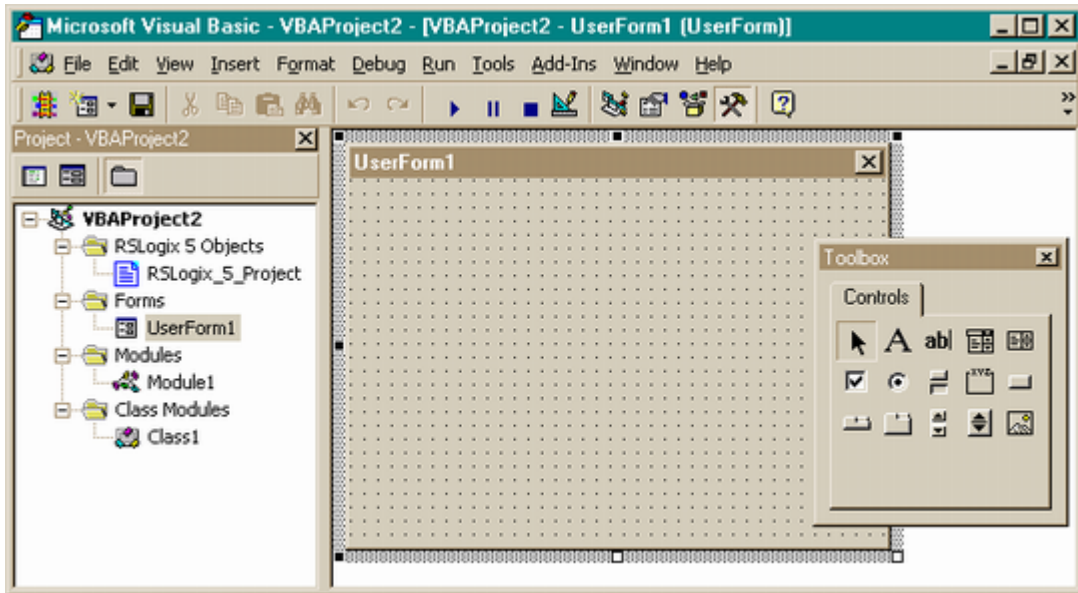
Since VBA contains a full Visual Basic implementation, including a project space, full language syntax, debugging, the forms package with ActiveX Controls, and an Object browser, it can save you money over purchasing a separate stand-alone copy of VB.

Solutions created with VBA execute quickly, since they run in the same memory space as the host application and are tightly integrated with it. Such quick execution allows developers to write code that responds to user actions, such as opening, closing, or saving projects, or reaching database information through code.

If you use products with embedded VBA the programming environment, including language, interface, and so on, is truly standard no matter which of the VBA licensed applications are involved. For example, the standard applies whether you use Rockwell Software’s RSLogix 5™, RSLogix 500™, or RSVIEW32™ or even Microsoft® Excel or Word.

Object models also mean a more open environment. If there are many vendors producing VBA-enabled applications, the walls of proprietary technology barriers start to break down. Therefore, developers building multiple-application software solutions can concentrate more on the functionality of the application, instead of wasting time and resources trying to get the different vendors packages to communicate or share data.

The VBA environment, shown below in the RSLogix5 software product, is the same everywhere it appears.



Uses

VBA uses the objects, methods, properties and events of the RSLogix automation interface to enable you to author scripts to automate tasks *within* the RSLogix editor. You can automate many of the routine, repetitive tasks involved in setting up RSLogix projects and customize your application. For example using the RSLogix object model you can automate functionality from within the Ladder Logic editor. Some uses might include:

- generating RSLogix ladder logic template files from code libraries automating your project creation tasks
- building individualized interfaces that execute functionality within RSLogix geared to specific groups within the factory environment, essentially “wrapping” subsets of functionality in your own interface
- generating event-driven HTML reports

- connecting your project to a web server so that it can be viewed over the internet
- tying applications together so developers can share application data and functionality within a common environment

Finding your way around this book

You'll see that the chapters in this book are organized by objects, starting with those basic to general file creation and ladder logic editing. Read the tables that follow for a chapter-by-chapter summary of the information in this book.

Automating the ladder logic editor

The following objects represent those functional areas that relate to file creation and manipulation and to the graphical ladder logic that defines your control program.

Objects:	Purpose:	Chpt:
Application	The Application object represents the RSLogix application. Use it to get other objects and perform top-level operations. (Make sure to read the important programming advice on Page 10 when using the Application object from within VBA.)	2
LogixProject	The LogixProject object represents the RSLogix project. Use it to access, define and return various attributes of an existing RSLogix project.	3
Processor	The Processor object represents the PLC, SoftLogix, SLC or MicroLogix processor. Use it to automate online functionality such as enabling and disabling forces, changing properties or handling edits.	4
ProgramFiles	The ProgramFiles collection represents all the program files in the project. Use it to add or remove program files from a collection.	5
ProgramFile	The ProgramFile object represents base functionality of a program file. Use it to return file characteristics such as online and protection status or use it to name the file.	6

Objects:	Purpose:	Chpt:
DataFiles	The DataFiles collection represents all the data files in the project. Use it to add or remove data files from a collection or read raw data.	7
DataFile	The DataFile object represents a data file in the project or processor. Use it to return defined attributes of the file.	8
LadderFile	The LadderFile object represents a ladder file in the project/processor. Use it to learn a file's attributes or manipulate rungs in the ladder file.	9
Rung	The Rung object represents a rung of ladder logic. Use it to obtain information about the rung.	10
RevisionNotes	The RevisionNotes object contains the revision notes for the project. Use it to get an indexed revision note or return the number of notes recorded for the project.	11
ReportOptions	The ReportOptions object represents the report settings associated with the project. Use it to read, establish or change settings.	12
PasswordPrivilegeConfig	The PasswordPrivilegeConfig object represents the master and class privilege administration unique to RSLogix5.	17

Automating the documentation database editor

RSLogix 5 and RSLogix 500 (Professional) versions 5.50 and later provide full database functionality with the following objects and collections.

Objects:	Purpose:	Chapter:
AddrSymRecords	The AddrSymRecords collection represents all records in the address/symbol editor list. Use it to add or remove entries from the collection or read raw data.	13
AddrSymRecord	The AddrSymRecord object represents data in the address/symbol editor list. Use it to return or set a value in any field.	14

Objects:	Purpose:	Chapter:
RungCmntPageTitleRecords	The RungCmntPageTitleRecords collection represents all records in the rung comment/page title editor list. Use it to add or remove entries from the collection or read raw data.	15
RungCmntPageTitleRecord	The RungCmntPageTitleRecord object represents data in the rung comment/page title editor list. Use it to return or set a value in any field.	16

Supplemental information

The appendices in this book provide this additional information.

Title:	Purpose:	Appendix:
Object Model Diagrams	View complete object model diagrams for RSLogix 5 and RSLogix 500.	A
Valid Type Definitions	Descriptive lists detailing how to construct valid terms of a type. Separate lists are included for RSLogix 5 and RSLogix 500.	B
Handling Errors	A discussion and brief example of how to programmatically handle exceptions thrown by RSLogix.	C
General Differences in the RSLogix 5 and 500 Automation Interfaces	Summary table of the differences in the object models of RSLogix 5 and RSLogix 500	D

Example files

Most chapters in this book include an example to help you understand how to use the object model. Although written for the RSLogix 5 software product, they may be easily adapted to RSLogix 500. For example type definitions may vary between products, and those differences must be considered when adapting code to the RSLogix 500 object model.

To assist you as you write your code, some samples are included on your RSLogix product compact disk. Look for the *VBA* samples (.rsp project) on the product CD to access these examples electronically.

How to access VBA in RSLogix 5 and RSLogix 500

To access VBA from within an RSLogix application follow these simple steps:

1. Open the project in your RSLogix software to which you want to attach your Visual Basic code.
2. Press [Alt]+[F11] to open the Microsoft Visual Basic project window.
3. Press [F2] to call the Object Browser. Make sure that the respective RSLogix5 or RSLogix500 type library is listed in the type library index.
4. The Project Explorer window (pictured in the left pane in the illustration on Page 2) displays forms, modules (files that hold the supporting code for the application), classes (advanced modules), and more. When you want to work with a particular part of the loaded application, double-click the component in the Project Explorer window to bring that component into focus.

Create your VBA code

Using the Visual Basic programming language, create subroutines in the code window for the project. Create subroutines that work with the RSLogix 5 or RSLogix 500 objects.

VBA subroutines run on a first-in, first-out basis. Each subroutine runs to completion before the next subroutine is started. For that reason, do not create subroutines that wait for user input before proceeding because if a user does not respond, all processing of subroutines stops. For example, if you create a dialog box that requires user input and no one responds to that dialog box, all processing of subroutines stops until the user input is received (although your RSLogix 5 or RSLogix 500 software continues to run normally).

If you want to turn off events in VBA, turn Design Mode ON. To do this click **Tools > Visual Basic > Design Mode**.

Some quick programming tips

The key to productive development using the RSLogix automation interface is a solid understanding of the methods, properties, and events that make RSLogix programmable – in other words, you need to understand the RSLogix object model.

Additionally, keep the following in mind while you code your customized applications:

1. Every method that accesses a COM object should have an On Error Resume Next or On Error Goto statement at the beginning of the method. Check for errors by testing the ErrObject often. Some of the typical errors that may occur are:
 - Object not set (a previous call that returned an object was not successful)
 - Method not supported on this object (maybe a spelling error, or you are using the wrong object)
2. If you receive a message: “Client has disconnected from Server,” that means that RSLogix has severed the link between the Visual Basic operation and itself. You will have to re-establish the link.
3. When using VBA the topmost object is the LogixProject object. This means that using the gApplication.Upload method exits the current project and displays the uploaded project. Any code you may have written stays with the exited project.
4. Use Visual Basic’s With statement to access an object’s properties to improve performance. Consider this RSLogix 5 example:

```
Dim Proc As RSLogix5.Processor
Set Proc = myProject.Processor
If Not Proc Is Nothing Then
    With Proc
        Revision.Text = .Revision
        Series.Text = .Series
        Subrev.Text = .Subrevision
    End With
End If
```
5. Don’t forget to use the Set statement for assigning object references.
6. Watch out for the proper use of parentheses in function calls. Overuse of parentheses may cause Visual Basic to evaluate an object, rather than passing its reference as an argument.

Chapter
2

Application object

The Application object represents the RSLogix application. This is the topmost object used to get other objects and perform top level operations. To use Automation to control RSLogix from another application, use the `CreateObject("RSLogix5.Application")` function to return an RSLogix 5 Application object or `CreateObject("RSLogix500.Application")` function to return an RSLogix 500 Application object. The Application Object is a creatable object.

Properties	Methods	Events
Application	FileNew	AfterUpload
AutoSaveInterval	FileOpen	BeforeFileNew
BackupCount	GetActiveProject	BeforeFileOpen
EnableAutoArrange	GetProcessorTypes	BeforeOffline
EnableAutoSave	GoOffline	BeforeOnline
EncodedRouteString	GoOnline	BeforeUpload
FullName	Quit	ClosingAllProjects
LibrarySearchPath	Upload	Quit
MaxDescriptionLineLength		
MaxSymbolLength		
Name		
NumberOfDescriptionLines		
Parent		
PromptForRevNote		
ProVersion		
SourceSearchPath		
VBAVersion		
VBE		
Version		
Visible		
WindowHandle		
WindowState		

Important

When using the Application Object from within VBA:

- The Create Object function does not apply.
 - To use Application from VBA you must access the LogixProject.Application property.
 - When using VBA the topmost object is the LogixProject object. This means that using any method that closes the current project will cause the code you've written to remain with the closed project. For example the gApplication.Upload method exits the current project and displays the uploaded project. Any code you may have written stays with the exited project.
-

The following commented code example illustrates how you might use the Application object to open an instance of RSLogix 5. The example further adds error checking and displays a message if the RSLogix 5 application could not be found.

```
'start Logix and store it in the gApplication object
Public Function StartLogix()
Set gApplication = CreateObject("RSLogix5.Application")
If gApplication Is Nothing Then
'Error checking, if gApplication is not set then display a message
MsgBox "ERROR: Failed to create gApplication Object, Logix
could not be started.", vbExclamation, "ERROR: 001"
End If
End Function
```

Properties










In most cases properties are characteristics or attributes of an object. Using a property returns information about the object or causes a quality of the object to change. The following properties, listed alphabetically, affect the appearance of the Application object by defining the way it should look or act.















Application

Application - Read Only

Used without an object qualifier, this property returns an Application object that represents the RSLogix application. Used with an object qualifier, this property returns an Application object that represents the class of the specified object (you can use this property with an OLE Automation object to return that object's application).

	AutoSaveInterval	Long - Read/Write
Gets or sets the way that RSLogix handles the interval of the auto save (in minutes). Use this for automatic file recovery when the project is not properly closed.		
	BackupCount	Long - Read/Write
Returns the current backup count.		
	EnableAutoArrange	Boolean - Read/Write
Returns or sets the way that RSLogix handles the re-arranging of the windows and results window when a verify or search all is performed.		
	EnableAutoSave	Boolean - Read/Write
Gets or sets the AutoSave feature in RSLogix. (1) indicates that the autosave feature is enabled. To ensure that the autosave feature initiates properly always save any new file immediately after creating it.		
	EncodedRouteString	String - Read/Write
Internal use only.		
	FullName	String - Read Only
The full name of the application.		
	LibrarySearchPath	String - Read/Write
The path used for library files. This path should not exceed 256 characters.		
	MaxDescriptionLineLength	Long - Read/Write
Gets or sets the default length for descriptions used by the RSLogix database.		
	MaxSymbolLength	Long - Read/Write
Gets or sets the default symbol length to be used by the database. By default, when you use RSLogix as the database editor symbols can be up to 20 characters in length. You can, however set the symbol length to 10 or 15 characters.		

 Name	String - Read Only
The name of the application: “RSLogix 5” or “RSLogix 500.”	
 NumberOfDescriptionLines	Long - Read/Write
Gets or sets the default number of lines that RSLogix will accept for a description in its database.	
 Parent	Application - Read Only
Returns the parent of the Application object. This represents the entire RSLogix application.	
 PromptForRevNote	Boolean - Read/Write
Gets or sets the display of the revision note dialog when doing a “Save” or “Save As” operation in RSLogix.	
 ProVersion	Boolean - Read Only
Returns if this version of RSLogix is the “Pro” version or the “Standard” version.	
 SourceSearchPath	String - Read/Write
Gets or sets the path used for the searching of source projects. This path is used when going online, uploading, opening, and saving.	
 VBAVersion	String - Read Only
Returns the version of the VBA software currently running.	
 VBE	Object - Read Only
Returns the VBA IDE extensibility object. The integrated development environment (IDE) includes many of the elements familiar to developers using Visual Basic. An enhanced Visual Basic Editor (VBE), for example, now exists outside the host application in a separate window. As a result, developers write code in VBA and simultaneously review their programming in the host application. The VBE also provide enhanced tools for tracking projects, debugging, setting priorities and protecting project code.	

	Version	String - Read Only
Returns the RSLogix version number in a text format.		
	Visible	Boolean - Read/Write
Gets or sets the visibility of the application. This property must be set if you plan to use methods that show dialogs within the RSLogix application.		
	WindowHandle	Long - Read Only
The window's handle to the application's main window.		
	WindowState	IgxWindowStateConstants - Read/Write
Gets or sets the state of the main application window. These states are valid:		
<ul style="list-style-type: none"> ▪ (0) IgxWindowStateNormal - The display window is in its normal state. ▪ (1) IgxWindowStateMinimized - The display window has been minimized to an icon. ▪ (2) IgxWindowStateMaximized - The display window has been enlarged to maximum size 		

Methods

Using a method causes something to happen to an object. In most cases methods are actions. After having initialized the RSLogix application object, use any of the following methods to identify the action that the object can perform. Although written for the RSLogix 5 software product, the short examples following each method may be easily adapted to RSLogix 500. For example, type definitions may vary between products, and those differences must be considered when adapting code to the RSLogix 500 object model.

	FileNew	LogixProject
Use this method to create a new RSLogix project.		

Syntax

FileNew(ProcessorType as IgxProcessorTypeConstants, IgnorePrompts as Boolean, SaveChanges as Boolean) as LogixProject

Arguments

ProcessorType - When this argument is set to IgxUnknownProc the processor selection dialog is displayed, otherwise if a valid processor type is supplied no dialog is displayed. A complete list of valid type definitions is in Appendix B.

IgnorePrompts - If set to True no user interface prompts are displayed to the user. If False prompts are displayed.

SaveChanges - If set to True any changes to the current open document are saved. If False changes to the current open document are not saved. This parameter is ignored if IgnorePrompts is set to False.

Returns

If successful the newly created LogixProject object is returned otherwise "Nothing" is returned.

Example

The following sample makes the call to RSLogix to make a new project using the parameters specified.

```
Set gLogixProject = gApplication.FileNew(lgxPLC_580E, True, False)
```



FileOpen

LogixProject

Use this method to open an existing RSLogix project.

Syntax

FileOpen(PathName as String, ShowDialog as Boolean, UseAutoSave as Boolean, AutoImportDB as Boolean) as LogixProject

Arguments

PathName - The string passed in this argument should be a fully qualified path name.

ShowDialog - If no user interface is desired set this to False.

UseAutoSave - Set this flag True to use an auto-recovery file (if it is present) when opening the file.

AutoImportDB - If True an archive file that was created with AI or APS will have the database automatically imported.

Returns

If successful the newly created LogixProject object is returned otherwise "Nothing" is returned.

Example

The following example makes the call to RSLogix to open the file named "Temp" at the path indicated, and include an import of the Database.

```
Set gLogixProject = gApplication.FileOpen("D:\RSI\Projects\Temp.rsp",  
False, False, True)
```

 **GetActiveProject** **LogixProject**

Use this method to get the current RSLogix Project.

Syntax

GetActiveProject()As LogixProject

Returns

Returns the current active project.

Example

The following example gets the active project from the application object.

```
Set gLogixProject = gApplication.GetActiveProject
```

 **GetProcessorTypes** **Long**

Use this method to get the list of supported processor types.

Syntax

GetProcessorTypes(*TypesArray as Variant, DescArray as Variant*) as Long

Arguments

TypesArray - The integer values for each of the enumerated types.


DescArray - The string values of each of the enumerated types.

Returns

If successful the length of both arrays is returned.

Example

```
Length = gApplication.GetProcessorTypes (TypeNums, TypeStrings)
```

 **GoOffline** **LogixProject**

Use this method to go offline with the processor.

Syntax

GoOffline(*IgnorePrompts as Boolean, SaveChanges as Boolean,*
[OnlineFileAction as lgxUpDownloadAction,] [PathName as String])
as LogixProject

Arguments

IgnorePrompts - If TRUE no user interface prompts, questions or warnings are displayed.

SaveChanges - If True changes are saved. If False changes are not saved. This parameter is ignored if IgnorePrompts is set to False.

OnlineFileAction - This is optional and will not affect the operation of the method.

PathName - This is optional and will not affect the operation of the method.

Example

The following example takes a project offline after saving without prompting the user.

```
Set gLogixProject = gApplication.GoOffline(True, True)
```



GoOnline

LogixProject

Use this method to go online with the processor.

Syntax

```
GoOnline(IgnorePrompts as Boolean, SaveChanges as Boolean,  
[OnlineFileAction as lgxUpDownloadAction,] [PathName as String])  
as LogixProject
```

Arguments

IgnorePrompts - If True no user interface prompts, questions or warnings are displayed.

SaveChanges - If True changes are saved. If False changes are not saved. This parameter is ignored if IgnorePrompts is set to False.

OnlineFileAction - [optional] This can be either (1) lgxUploadCreateNew, (2) lgxUploadCurrent or (3)lgxUploadPath. This parameter is ignored if IgnorePrompts is set to False.

PathName - [optional] The fully qualified path of the file, only used with lgxUploadPath. This parameter is ignored if IgnorePrompts is set to False.

Example

The following example takes the current project online after saving without prompting the user.

```
Set gLogixProject = gApplication.GoOnline(True, True)
```

Quit

Use this method to quit RSLogix.

Syntax

Quit(IgnorePrompts as Boolean, SaveChanges as Boolean)

Arguments

IgnorePrompts - If True no user interface prompts, questions or warnings are displayed.

SaveChanges - If True changes are saved. If False changes are not saved. This parameter is ignored if IgnorePrompts is set to False.

Example

The following example quits without saving and without prompting.

Call `gApplication.Quit(True, False)`

Upload

LogixProject

Use this method to upload the processor program into the current project.

Syntax

Upload(IgnorePrompts as Boolean, SaveChanges as Boolean, Upload.Action as lgxUpDownloadAction, Online.Action as lgxOnlineAction, [PathName as String]) As LogixProject

Arguments

IgnorePrompts - If True no user interface prompts, questions or warnings are displayed.

SaveChanges - If True changes are saved. If False changes are not saved. This parameter is ignored if IgnorePrompts is set to False.

Upload.Action - The flag UploadAction, which is ignored if IgnorePrompts is set to False can be one of the following indicating where the project it is uploading to:

- (1) lgxUploadCreateNew
- (2) lgxUploadCurrent
- (3) lgxUploadPath

Online.Action - Places the processor in the selected mode of operation. This can be either (1) lgxGoOnline or (2) lgxGoOffline. This parameter is ignored if IgnorePrompts is set to False.

PathName - [optional] The fully qualified path for the file to be created or the specified path and filename to go online with. This parameter is only used with `lgxUploadPath`. This parameter is ignored if `IgnorePrompts` is set to `False`.

Example

The following example uploads the current project from the current processor without prompting or saving the changes that were made and then going offline.

```
Set gLogixProject = gApplication.Upload(True, False, lgxUploadCurrent, lgxGoOffline)
```

Events

We recommend that you first set up an event class module to catch events. When an instance of the class is created, you can apply these events to the `Application` object. The following code example illustrates how to set up an event class for an RSLogix5 application.

1. Create a new class module.
2. Then connect the `Application` object in your main code to the class.

```
Dim WithEvents gAppEvents As RSLogix5.Application
```

```
Public Sub ConnectToEvents (pApp As RSLogix5.Application)  
    Set gAppEvents = pApp  
End Sub
```

AfterUpload

Syntax

```
AfterUpload()
```

Remarks

This event is raised when the upload has finished via automation or any way from the application.

Example

The following example is simple debug code that outputs the message “Upload Finished” to confirm the event was called.

```
Private Sub gAppEvents_AfterUpload()  
    Debug.Print ("Upload Finished")  
    'output a message to the user confirming the event was called  
End Sub
```



BeforeFileNewBoolean

Syntax

BeforeFileNew() As Boolean

Remarks

This event is raised when the FileNew action is invoked via automation or any way from the application. If this event returns True the action is aborted. If this event returns False the action continues.

Example

The following example is simple code to output the message “Cannot Create New File” and abort the action of creating a new file.

```
Private Function gAppEvents_BeforeFileNew() As Boolean
    'Display message explaining that this operation is not permitted
    MsgBox ("Cannot Create New File")
    'Return a value of True to cancel the operation
    gAppEvents_BeforeFileNew = True
End Function
```



BeforeFileOpenBoolean

SyntaxBeforeFileOpen(*FileName as String*) As Boolean**Remarks**

This event is raised when the FileOpen action is invoked via automation or any way from the application. FileName is the fully qualified path of the file to be opened. If this event returns True the action is aborted. If this event returns False the action continues.

Example

The following example is simple code that outputs the message “Opening File” once the event is called.

```
Private Function gAppEvents_BeforeFileOpen(ByVal Filename As String)
As Boolean
    'Display a message confirming that the event was called
    MsgBox ("Opening File")
    'Return a value of False to proceed with the operation
    gAppEvents_BeforeFileOpen = False
End Function
```



BeforeOffline**Boolean**

Syntax

BeforeOffline() As Boolean

Remarks

This event, raised before the action of going from online to offline, is invoked via automation or any way from the application. If this event returns TRUE the action is aborted, if FALSE is returned the action will continue.

Example

The following example is simple code to output the message “Going Offline” when the BeforeOffline event is called.

```
Private Function gAppEvents_BeforeOffline() As Boolean
    'Display a message confirming that the event was called
    MsgBox ("Going Offline")
    'Return a value of False to proceed with the operation
    gAppEvents_BeforeOffline = False
End Function
```



BeforeOnline**Boolean**

Syntax

BeforeOnline() As Boolean

Remarks

This event is raised before the action of going from offline to online is invoked via automation or any way from the application. If this event returns FALSE the action is aborted, if TRUE is returned the action will continue.

Example

The following example is simple code to output the message “Cannot Go Online” when the BeforeOnline event is called.

```
Private Function gAppEvents_BeforeOnline() As Boolean
    'Display a message explaining that this operation is not permitted
    MsgBox ("Cannot Go Online")
    'Return a value of True to cancel the operation
    gAppEvents_BeforeOnline = True
End Function
```



BeforeUpload**Boolean**

Syntax

Before Upload() As Boolean

Remarks

This event is raised before the Upload action is invoked, via automation or any way from the application. If this event returns TRUE the action is aborted, if FALSE is returned the action will continue.

Example

The following example is simple code that outputs a message confirming that the Upload is proceeding.

```
Private Function gAppEvents_BeforeUpload() As Boolean
    'Display a message confirming that the event was called
    MsgBox ("Uploading")
    'Return a value of False to proceed with the operation
    gAppEvents_BeforeUpload = False
End Function
```



ClosingAllProjects

Syntax

ClosingAllProjects()

Remarks

This is an application level event raised any time a project is closed.

Example

The following example is simple debug code that outputs the message “RSLogix 5 closing all opened projects” to confirm the event was called.

```
Private Function gAppEvents_ClosingAllProjects()
    Debug.Print ("RSLogix 5 closing all opened projects")
    'output a message to the user confirming the event was called
End Function
```



Quit

Syntax

Quit()

Remarks

This event is raised when the application is ready to shutdown.

Example

The following simple debug code outputs the message “RSLogix 5 Exiting” to confirm the event was called.

```
Private Sub gAppEvents_Quit()  
    Debug.Print ("RSLogix 5 Exiting")  
End Sub
```

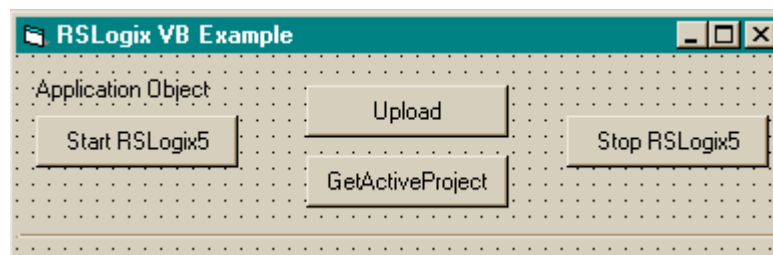
Summary example

Important This book assumes that you have the basic knowledge required to work with forms and controls in Visual Basic.

The following example automates some top level actions that can be accomplished using the Application object’s properties, methods and events in the RSLogix automation interface. Comments within the code are preceded by an apostrophe ('). You’ll see that although the example is specific to RSLogix 5 software, it is generic enough to adapt to RSLogix 500 with only minor form and comment alterations.

Form

In subsequent chapters throughout this manual the following basic form will be added to as the complete functionality of the automation interface is introduced object by object.



Code

```
'-----  
' Global variables  
'-----  
  
Dim gApplication As RSLogix5.Application 'Application object  
Dim gProject As RSLogix5.LogixProject 'LogixProject object  
  
'-----  
' Application Object  
'-----  
  
Private Sub Command1_Click()  
  
    ' Set the application object to the object returned by CreateObject.  
    ' CreateObject is simply a method provided by Microsoft that creates  
    ' a new registered COM application instance. In this case we start  
    ' RSLogix 5 by using the "RSLogix5.Application" string.  
    Set gApplication = CreateObject("RSLogix5.Application")  
    ' At this point, if the CreateObject method functioned properly, the  
    ' gApplication object is now a direct reference to the RSLogix5  
    ' Object Model. Any properties or methods that we invoke on this  
    ' object will immediately take effect in RSLogix.  
  
    ' Immediately set the visible property of the application to 'True'  
    gApplication.Visible = True  
  
    ' Assign the AutoSaveInterval value to 3 minutes  
    gApplication.AutoSaveInterval = 3  
  
    ' Assign WindowState prop. to lgxWindowStateMaximized enumeration  
    gApplication.WindowState = lgxWindowStateMaximized  
  
End Sub  
  
Private Sub Command2_Click()  
    ' Quit the application ignoring prompts and not saving changes.  
    gApplication.Quit True, False  
    ' Eliminate the reference to the application object  
    Set gApplication = Nothing  
End Sub  
  
Private Sub Command3_Click()  
On Error GoTo errorHandler  
    ' Upload a project from the processor using the upload method of the  
    ' application object while ignoring prompts, NOT saving the previous  
    ' file, creating new file from the upload (using lgxUploadCreateNew  
    ' enum (see the objectbrowser for more enumerations)), and go online  
    ' (using the lgxGoOnline enum).  
    ' Set the returned object reference to the gProject object.
```

```

        Set gProject = gApplication.Upload(True, False, lgxUploadCreateNew,
lgxGoOnline)
Exit Sub

errorHandler:
    ' Upon a caught error decide what to do.
    MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Command4_Click()
    ' Get the currently open project in the application.
    Set gProject = gApplication.GetActiveProject
End Sub

```


Chapter
3

LogixProject object

The LogixProject object represents the RSLogix project. The LogixProject Object can only be obtained from the Application Object via any one of the following methods:

- FileNew
- FileOpen
- GetActiveProject
- GoOffline
- GoOnline
- Upload

You cannot create a new instance of a LogixProject object with the CreateObject function.


Properties	Methods	Events
AddrSymRecords	Close	AfterDownload
Application	DisplayReportOptions	AfterOpen
DataFiles	Download	AfterSave
FullName	GotoDataFileElement	BeforeClose
Modified	GotoProgramFile	BeforeDownload
Name	ImportDataBase	BeforeSave
Online	PrintReport	BeforeSaveAs
Parent	Save	FinishedReport
PasswordPrivilegeCfg (5 only)	SaveAs	FinishedVerify
Processor	ShowControllerProperties	OnlineOfflineFileClosing
ProgramFiles	ShowDataFile	
ReportOptions	ShowDataTablesProperties	
Revision	ShowProgramFile	
RevisionNotes	ShowProgramFilesProperties	
RungCmntPageTitleRecords	VerifyProject	
	VerifyProgramFile	

The following commented code example illustrates how you might open a project after first determining whether a project is already opened. The example first closes any opened project, waits for the project to close, and then opens the project named in the Filename parameter.


```
'This function opens a file for use in RSLogix5, it stores the
'LogixProject object in gLogixProject
Public Function OpenAFile(Filename As String)
If Not gLogixProject Is Nothing Then
'if a file is open close it
  gLogixProject.Close True, True
  'call RSLogix to close project
  Form1.Timer2.Enabled = True
  'wait for a full second, this is so the project has time to close
  While (Form1.Timer2.Enabled = True)
    DoEvents
    'Timer will disable itself after one second
  Wend
  Set gLogixProject = Nothing
  'clear the gLogixProject object
End If
Set gLogixProject = gApplication.FileOpen(Filename, False, False, True)
'make call to RSLogix to open the File passed into this function
If gLogixProject Is Nothing Then
'if the above call failed then display a message and exit
  MsgBox "Logix failed to create the project!", vbExclamation, "ERROR"
  Exit Function
End If
End Function
```

Properties












In most cases properties are characteristics or attributes of an object. Using a property returns information about the object or causes a quality of the object to change. The following properties define the LogixProject object

 AddrSymRecords	AddrSymRecord - Read Only
---	----------------------------------

Returns the AddrSymRecords collection.

 Application	Application - Read Only
--	--------------------------------

Used without an object qualifier, this property returns an Application object that represents the RSLogix application.

	DataFiles	DataFiles - Read Only
	Returns the data files collection object/interface.	
	FullName	String - Read Only
	The full name of the project including the fully qualified path.	
	Modified	Boolean - Read Only
	Indicates if the project has been modified in any way.	
	Name	String - Read Only
	The name of the project.	
	Online	Boolean - Read Only
	Returns whether or not RSLogix is online with the processor.	
	Parent	Application - Read Only
	Returns the object representing the entire RSLogix Application.	
	PasswordPrivilegeCfg	PasswordPrivilegeCfg - Read Only
	Returns the Password/Privilege configuration for the processor.	
	Processor	Processor - Read Only
	Returns the processor object.	
	ProgramFiles	ProgramFiles - Read Only
	Returns the program files collection object.	
	ReportOptions	ReportOptions - Read Only
	Returns the report options object/interface.	
	Revision	Integer - Read Only
	Returns the current revision of the project.	

(5 only)



RevisionNotes**RevisionNotes - Read Only**

Returns the RevisionNotes collection.



RungCmntPageTitleRecords**RungCmntPageTitleRecords - Read Only**

Returns the RungCmntPageTitleRecords collection.

Methods

Using a method causes something to happen to an object. In most cases methods are actions. Use any of the following methods to identify an action for the LogixProject object to perform. Although written for the RSLogix 5 software product, the short examples following each method may be easily adapted to RSLogix 500. For example, type definitions may vary between products, and those differences must be considered when adapting code to the RSLogix 500 object model.



Close

Use this method to close the RSLogix project.

Syntax

Close(IgnorePrompts as Boolean, AcceptDefaultActions as Boolean)

Arguments

IgnorePrompts - When True no user interface confirmations will be displayed. If FALSE prompts are displayed.

AcceptDefaultActions - If True the default saving action that had been selected for the project is followed. This parameter is ignored if IgnorePrompts is set to False.

Example

The following code snippet makes the call to RSLogix to close the project after first saving but without user prompts.

```
Call gLogixProject.Close(True, True)
```



DisplayReportOptions

Use this method to display the Report Options dialog.

Syntax

DisplayReportOptions()

Returns

If successful the Report Options dialog is displayed.

Example

The following code snippet displays the Report Options dialog for the user.

```
gLogixProject.DisplayReportOptions
```



Download

Boolean

Use this method to download the current project into the processor.

Syntax

Download(*IgnorePrompts as Boolean, OnlineAction as lgxOnline.Action, ProcessorMode as lgxProcOnline.State*) As Boolean

Arguments

IgnorePrompts - When True no user interface prompts, questions or warnings will be displayed. If False prompts are displayed.

OnlineAction - This can be either (1) lgxGoOnline or (2) lgxGoOffline, indicating what to do after a successful download. This parameter is ignored if IgnorePrompts is set to False.

ProcessorMode - This can be either (6) lgxRemoteProg, (7) lgxRemoteTest or (8) lgxRemoteRun and only applies if lgxOnlineAction = lgxGoOnline. This parameter is ignored if IgnorePrompts is set to False.

Remarks

To download the processor must be in PROGRAM mode and the key switch must be in either REM or PROG. You must also be offline with RSLogix.

Returns

If successful a value of True is returned; if not successful False is returned.

Example

The following code snippet calls RSLogix to download the project without prompting; then go online and place the processor in Remote Run mode.

```
Result = gLogixProject.Download(True, lgxGoOnline, lgxREMOTERUN)
```



GotoDataFileElement

Boolean

Use this method to display the indicated data file element.

Syntax

GotoDataFileElement(*Address as String*) As Boolean

Arguments

Address - The data file address you want displayed.

Returns

If successful a value of True is returned, and the selected data file is displayed with the selected address/element highlighted. If unsuccessful False is returned.

Example

The following code snippet displays the data file after having been passed the address via string input from a text box on a form.

```
Result = gLogixProject.GotoDataFileElement(Text1.Text)
```



GotoProgramFile

Boolean

Use this method to display the indicated program file.

Syntax

GotoProgramFile(*FileNumber as Long, RungNumber as Long,
Ins as Long, Op as Long*) as Boolean

Arguments

FileNumber - The number of the desired file.

RungNumber - The number of the desired rung.

Ins - The instruction you want to display.

Op - The operand index for the desired instruction.

Returns

If successful a value of True is returned, the selected program file displays and the instruction operand selected is highlighted. If unsuccessful False is returned.

Example

The following example goes to the program file 2 and highlights rung 1. By indicating a 0 for the instruction number in the third argument, the cursor assumes position before the first instruction on the rung.

```
Call gLogixProject.GotoProgramFile(2, 1, 0, 1)
```



ImportDataBase**Boolean**

Use this method to import the RSLogix 5 or RSLogix 500 documentation database information from a CSV (comma separated variable) or EAS (Exported symbol/description file) text file. This will overwrite existing database information. Currently only Address Symbol database imports are supported.

Syntax

ImportDataBase(PathName as String, ShowDialog as Boolean, [DBImportType as lgxImportDBTypes]) as Boolean

Arguments

PathName - The fully qualified path for the text file to import.

ShowDialog - Enter True to show the Import Database dialog. If you choose to show this dialog, then any pathname provided in the PathName parameter is ignored.

DBImportType - Determines which documentation database is being imported.

Returns

True is returned if the import was successful. False is returned if the import was unsuccessful.

Example

This code snippet displays the Import Database dialog allowing you to proceed to select database files from the dialog.

```
Call gLogixProject.ImportDataBase("C:\Project\AddrSym.csv", True)
```



PrintReport**Boolean**

Use this method to print a report.

Syntax

PrintReport(IgnorePrompts as Boolean) As Boolean

Arguments

IgnorePrompts - If True no prompts, questions or warnings are displayed.

Returns

If successful a value of True is returned and a report is printed. If unsuccessful printing is cancelled and a value of False is returned.

Example

The following example makes a call to RSLogix to print the report.

```
Call gLogixProject.PrintReport(True)
```



Save

Boolean

Use this method to save a project.

Syntax

Save(IgnorePrompts as Boolean, AcceptDefaultActions as Boolean) As Boolean

Arguments

IgnorePrompts - If True no user interface prompts, questions or warnings are displayed.

AcceptDefaultActions - If True the default action proceeds. This parameter is ignored if IgnorePrompts is set to False.

Returns

If successful a value of True is returned and the LogixProject is saved as directed by the arguments in the call. If unsuccessful False is returned.

Example

The following code snippet is the call to save a project.

```
Call gLogixProject.Save(True, True)
```



SaveAs

Boolean

Use this method to save the project using a new name.

Syntax

SaveAs(IgnorePrompts as Boolean, AcceptDefaultActions as Boolean, DBAction as lgx.SaveAction, PathName as String) As Boolean

Arguments

IgnorePrompts - If TRUE no user interface prompts, questions or warnings are displayed.

AcceptDefaultActions - If True the default action proceeds. This parameter is ignored if IgnorePrompts is set to False.

DBAction - The lgxDBAction can be one of the following types:

- (0) lgxNoAction
- (1) lgxSaveNativeExternalDB
- (2) lgxSaveAIEExternalDB
- (3) lgxSaveAPSEExternalDB

These choices indicate the format that the database files will be exported to. This parameter is ignored if IgnorePrompts is set to False.

PathName - This is the fully qualified path/name of the new file/location to save the file. This parameter is ignored if IgnorePrompts is set to False.

Returns

If successful a value of True is returned and the LogixProject is saved as directed by the arguments in the call. If unsuccessful False is returned.

Example

The following code snippet saves the current project with the filename (Filename.rsp) without prompting.

```
Call gLogixProject.SaveAs(True, True, lgxNoAction,
"C:\FolderX\Filename.rsp")
```

ShowControllerProperties

Use this method to display the controller properties dialog.

Syntax

```
ShowControllerProperties()
```

Returns

When successful this displays the controller properties dialog.

Example

The following code snippet displays the controller properties dialog.

```
Call gLogixProject.ShowControllerProperties( )
```

ShowDataFile Boolean

Use this method to display a specific data file. The application's visible property must be set for this to work properly.

Syntax

```
ShowDataFile(File as Long) As Boolean
```

Returns

When successful a value of True is returned and the indicated data file is displayed. If unsuccessful False is returned.

Example

The following code snippet displays counter file (C5) for the current project.

```
Result = gLogixProject.ShowDataFile(5)
```



ShowDataTablesProperties

Use this method to display the data files' properties dialog.

Syntax

```
ShowDataTablesProperties()
```

Returns

When successful this displays the data files' properties dialog. It may be useful if you want to change the protection options placed on a particular data table file or change the file size.

Example

The following code snippet displays the properties dialog for data files.

```
Call gLogixProject.ShowDataTablesProperties ()
```



ShowProgramFile

Boolean

Use this method to display a program file. The application's visible property must be set for this to work properly.

Syntax

```
ShowProgramFile(File as Long) As Boolean
```

Returns

When successful a value of True is returned and the indicated program file is displayed. If unsuccessful False is returned.

Example

The following code snippet displays program file #3 in the current project.

```
Result = gLogixProject.ShowProgramFile (3)
```



ShowProgramFilesProperties

Use this method to display the program file's property dialog.

Syntax

```
ShowProgramFilesProperties()
```

Returns

When successful this displays the program files' properties dialog.

Example

The following code snippet displays the properties dialog for program files.

```
Call gLogixProject.ShowProgramFileProperties ()
```



VerifyProject

Boolean

Use this method to verify the RSLogix project and display the results.

Syntax

VerifyProject(*DisplayProgress as Boolean*) As Boolean

Arguments

Display Progress - If set to TRUE a dialog box displays the progress of the verify. If set to FALSE no user interface will be presented to the user indicating the progress of the verify operation. A results window will be shown at the end of the verify operation however.

Returns

When successful a value of True is returned and the project is verified and the results of the verify operation are displayed. If unsuccessful False is returned.

Example

The following code snippet calls for a project verification without displaying a dialog box to show how the verify is proceeding.

```
Result = gLogixProject.VerifyProject (False)
```



VerifyProgramFile

Boolean

Use this method to verify a designated program file.

Syntax

VerifyProgramFile(*FileNumber as Long*) As Boolean

Arguments

FileNumber - The number of the program file that is to be verified.

Returns

If the ladder file is verified without errors, True is returned, otherwise False is returned.

Example

The following code snippet makes the call to RSLogix to verify program file #2.

```
Result As Boolean  
Result = gLogixProject.VerifyProgramFile(2)
```

Events

We recommend that you first set up an event class module to catch events. When an instance of the class is created, you can apply these events to the LogixProject object.

The following code example illustrates how to set up an event class.

1. Create a new class module.
2. Next connect the LogixProject object in your main code to the class.

```
Dim WithEvents gProjEvents As RSLogix5.LogixProject

Public Sub ConnectToEvents(pProj As RSLogix5.LogixProject)
    Set gProjEvents = pProj
End Sub
```

AfterDownload

Syntax

AfterDownload()

Remarks

This event is raised at the end of the download action. This is just a notification event.

Example

The following example is simple debug code that outputs the message “Download Complete” to confirm the event was called.

```
Private Sub gProjEvents_AfterDownload()
    Debug.Print ("Download Complete")
    'output a message to the user confirming the event was called
End Sub
```

AfterOpen

Syntax

AfterOpen()

Remarks

This event is fired immediately after a project is opened and is used in VBA to perform initialization when a project is opened. It cannot be used in VB, however.

Example

The following example is simple debug code that outputs the message “File Opened” to confirm the event was called.

```
Private Sub gProjEvents_AfterOpen()  
    Debug.Print ("File Opened")  
    'output a message to the user confirming the event was called  
End Sub
```



AfterSave

Syntax

AfterSave()

Remarks

This event is raised at the end of the save action. This is just a notification event.

Example

The following example is simple debug code that outputs the message “File Saved” to confirm the event was called.

```
Private Sub gProjEvents_AfterSave()  
    Debug.Print ("File Saved")  
    'output a message to the user confirming the event was called  
End Sub
```



BeforeClose

Boolean

Syntax

BeforeClose() As Boolean

Remarks

This event is raised at the start of the close action. If the action is to be aborted return True otherwise return False to continue with the action.

Example

The following example is simple code that outputs the message “Closing Project” once the event is called and then proceeds with the operation.

```
Private Function gProjEvents_BeforeClose() As Boolean
    'Display a message confirming that the event was called
    MsgBox ("Closing Project")
    'Return a value of False to proceed with the operation
    gProjEvents_BeforeClose = False
End Function
```



BeforeDownload

Boolean

Syntax

BeforeDownload() As Boolean

Remarks

This event is raised at the start of the download action. If the action is to be aborted return True otherwise return False to continue with the action.

Example

The following example displays a message box advising of a problem with download.

```
Private Function gProjEvents_BeforeDownload() As Boolean
    'Display a message explaining that this operation is not permitted
    MsgBox ("Cannot Download")
    'Return a value of True to cancel the operation
    gProjEvents_BeforeDownload = True
End Function
```



BeforeSave

Boolean

Syntax

BeforeSave() As Boolean

Remarks

This event is raised at the start of the “Save” action. If the action is to be aborted return True, otherwise return False to continue with the action.

Example

The following example is simple code that outputs the message “Saving Project” to confirm the event was called.

```
Private Function gProjEvents_BeforeSave() As Boolean
    'Display a message confirming that the event was called
    MsgBox ("Saving Project")
    'Return a value of False to proceed with the operation
```

```
        gProjEvents_BeforeSave = False
    End Function
```



BeforeSaveAs

Boolean

Syntax

BeforeSaveAs() As Boolean

Remarks

This event is raised at the start of the “Save As” action. If the action is to be aborted return True, otherwise return False to continue with the action.

Example

The following example is simple code that outputs a message indicating inability to save a file under another name.

```
Private Function gProjEvents_BeforeSaveAs() As Boolean
    'Display a message explaining that this operation is not permitted
    MsgBox ("Cannot Save As Different File Name")
    'Return a value of True to cancel the operation
    gProjEvents_BeforeSaveAs = True
End Function
```



FinishedReport

Syntax

FinishedReport()

Remarks

This event is raised at the end of the print report action. This is just a notification event.

Example

The following example is simple debug code that outputs the message “Finished Printing Report” to confirm the event was called.

```
Private Sub gProjEvents_FinishedReport()
    Debug.Print ("Finished Printing Report")
    'output a message to the user confirming the event was called
End Sub
```

 **FinishedVerify**

Syntax

FinishedVerify()

Remarks

This event is raised at the end of the verify action. This is just a notification event.

Example

The following example is simple debug code that outputs the message “Verify Finished” to confirm the event was called.

```
Private Sub gProjEvents_FinishedVerify()  
    Debug.Print ("Verify Finished")  
    'output a message to the user confirming the event was called  
End Sub
```

 **OnlineOfflineFileClosing**

Syntax

OnlineOfflineFileClosing()

Remarks

This event is raised when the current open project is being closed when going from “online to offline” or “offline to online.”

Example

The following example is simple debug code that outputs the message “Closing File” to confirm the event was called.

```
Private Sub gProjEvents_OnlineOfflineFileClosing()  
    Debug.Print ("Closing File")  
    'output a message to the user confirming the event was called  
End Sub
```


Summary example

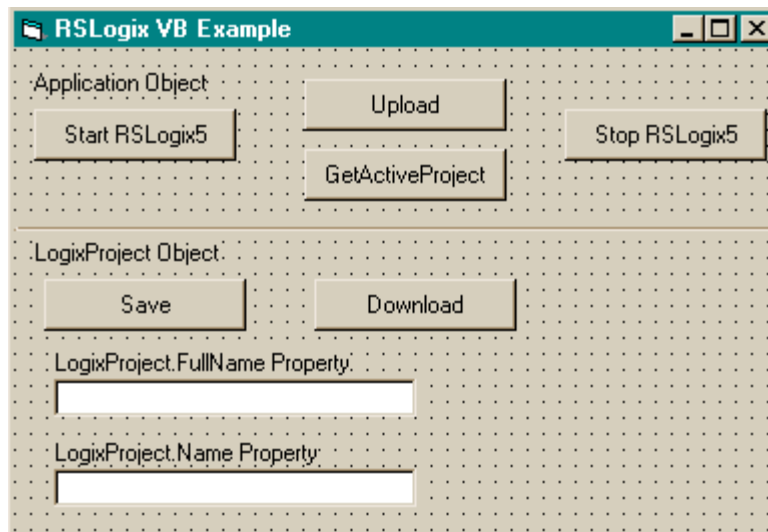
Important

This book assumes that you have the basic knowledge required to work with forms and controls in Visual Basic.

The following example automates functionality within RSLogix 5 with the automation interface by incorporating properties, methods and events from both the Application and LogixProject objects. Comments within the code are preceded by an apostrophe ('). You'll see that although the example is specific to RSLogix 5 software, it is generic enough to adapt to RSLogix 500 with only minor form and comment alterations.

Form

The following form builds on the form first presented in Chapter 2. Subsequent chapters in this book will continue to build on this form.



Code

```
'-----  
' Global variables  
'-----  
  
Dim gApplication As RSLogix5.Application 'Application object  
Dim gProject As RSLogix5.LogixProject 'LogixProject object
```

```

'-----
' Application
'-----

Private Sub Command1_Click()

    ' Set application object to the object returned from CreateObject.
    ' CreateObject is simply a method provided by Microsoft that creates
    ' a new registered COM application instance. In this case we start
    ' RSLogix 5 by using the "RSLogix5.Application" string.
    Set gApplication = CreateObject("RSLogix5.Application")
    ' At this point, if the CreateObject method functioned properly, the
    ' gApplication object is now a direct reference to the RSLogix5
    ' Object Model. Any properties or methods that we invoke on this
    ' object will immediately take effect in RSLogix.

    ' Immediately set the visible property of the application to 'True'
    gApplication.Visible = True

    ' Assign the AutoSaveInterval value to 3 minutes
    gApplication.AutoSaveInterval = 3

    ' Assign WindowState prop to lgxWindowStateMaximized enumeration
    gApplication.WindowState = lgxWindowStateMaximized

End Sub

Private Sub Command2_Click()
    ' Quit the application ignoring prompts and not saving changes.
    gApplication.Quit True, False
    ' Eliminate the reference to the application object
    Set gApplication = Nothing
End Sub

Private Sub Command3_Click()
    On Error GoTo errorHandler
    ' Upload a project from the processor using the upload method of the
    ' application object while ignoring prompts, NOT saving the previous
    ' file, creating new file from the upload (using lgxUploadCreateNew
    ' enum (see the objectbrowser for more enumerations)), and go online
    ' (using the lgxGoOnline enum).
    ' Set the returned object reference to the gProject object.
    Set gProject = gApplication.Upload(True, False, lgxUploadCreateNew,
    lgxGoOnline)
Exit Sub

errorHandler:
    ' Upon a caught error decide what to do.
    MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
    Err.Description

```

```

End Sub

Private Sub Command4_Click()
    ' Get the currently open project in the application.
    Set gProject = gApplication.GetActiveProject
End Sub

'-----
' LogixProject
'-----

Private Sub Command5_Click()
    Dim returnValue As Boolean

    ' Save the currently open project, assign the return value to a
    ' variable and display that value in a message box.
    returnValue = gProject.Save(True, True)
    MsgBox "Returned: " & returnValue

End Sub

Private Sub Command6_Click()
On Error GoTo errorHandler
    Dim returnValue As Boolean

    ' Download the project to the current processor. This method call is
    ' ignoring all prompts, going online (using the lgxGoOnline enum),
    ' setting the processor to remote program mode (using the
    ' lgxREMOTEPROG enum) and displaying the return value in a message
    ' box
    returnValue = gProject.Download(True, lgxGoOnline, lgxREMOTEPROG)
    MsgBox "Returned: " & returnValue
Exit Sub

errorHandler:
    ' Upon a caught error decide what to do.
    MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Text1_Click()
    ' Display the current name of the project in a text box.
    Text1.Text = gProject.FullName
    Text2.Text = gProject.Name
End Sub

```


Chapter
4

Processor object

The Processor object represents the processor being used in the current project. The Processor object is obtained from the LogixProject object via the Processor property. You cannot create a new instance of the Processor object with the CreateObject function.

Properties

Application
CanAssembleEdits
CanCancelEdits
CanTestEdits
CanUntestEdits
DefaultDriver
CurrentPLC5MemSize - *(RSLogix 5 only)*
DestNodeOctal
DriverName
DriverTimeout
EditsActive
EditsPresent
Emulator
EncodedRouteString
Faulted
HasPasswordPrivileges - *(RSLogix 5 only)*
KeySwitchPosition
Name
Node
NumberOfMemSizeChoices - *(RSLogix 5 only)*
Online
OnlineChangesMade
ProcessorMode
ProgramID - *(RSLogix 500 only)*
Revision - *(RSLogix 5 only)*
Series - *(RSLogix 5 only)*
SubRevision - *(RSLogix 5 only)*
Type

Methods

ClearAllForces
DisableForces
EnableForces
GetPLC5MemSizeChoiceByIndex - *(RSLogix 5 only)*
SetPLC5MemSize - *(RSLogix 5 only)*

Events

-None-

The following commented code example illustrates how you might establish which processor is being used in the current opened project. If no data is available, then an error message will be returned.

```
'get the processor object
Set gProc = gLogixProject.Processor
If gProc Is Nothing Then
'if that failed then exit
    MsgBox "Failed to get Processor Data from the LogixProject Object!",
vbExclamation, "ERROR 005"
    Exit Function
End If
```

Properties

In most cases properties are characteristics or attributes of an object. Using a property returns information about the object or causes a quality of the object to change. The following properties query the processor object for the stated information.



Application

Application - Read Only

This property returns an Application object that represents the RSLogix 5 or RSLogix 500 application.



CanAssembleEdits

Boolean - Read Only

Queries and returns whether or not edits can be assembled in the processor (incorporated into the ladder program while editing the ladder logic program online).



CanCancelEdits

Boolean - Read Only

Queries and returns whether or not edits can be cancelled in the processor.



CanTestEdits

Boolean - Read Only

Queries and returns whether or not edits can be tested in the processor (examine how the program operates with the edited rung).



CanUntestEdits

Boolean - Read Only

Queries and returns whether or not edits can be untested in the processor (return the operation of the program to the way it functioned before the edited rung was tested).

(5 only)



CurrentPLC5MemSize **Long - Read Only**

This property returns the returns the value of the current processor memory size in bytes.



DefaultDriver **Boolean - Read Only**

Queries and returns whether or not the default driver is being used in the current project.



DestNodeOctal **Boolean - Read Only**

Returns whether or not the destination node is “expressed in” or “expected to be in” octal.



DriverName **String - Read Only**

Returns the name of the communications driver currently being used to communicate with the processor.



DriverTimeout **Integer - Read/Write**

Returns or sets the timeout expressed in seconds for the communication driver.



EditsActive **Boolean - Read Only**

Queries and returns whether or not edits are active in the processor.



EditsPresent **Boolean - Read Only**

Queries and returns whether or not edits are present in the processor.



Emulator **Boolean - Read Only**

Queries and returns whether or not the emulator is being used instead of a real processor.



EncodedRouteString **String - Read/Write**

This property is for internal use only.



Faulted **Boolean - Read Only**

Queries and returns whether or not the processor is faulted.

(5 only)



HasPasswordPrivileges **Boolean - Read Only**

This property returns a Boolean which will be true if the processor type supports password privileges.



KeySwitchPosition **IgxKeyPositionConstants - Read Only**

Returns the current position of the key switch on the processor. Possible returned values are listed below and described in Appendix B. .

- (0) IgxUnknownKey
- (1) IgxKeyRemote
- (2) IgxKeyProgram
- (3) IgxKeyRun



Name **String - Read/Write**

Returns or sets the name of the processor.



Node **Integer - Read/Write**

Returns or sets the processor node number in decimal.

(5 only)



NumberOfMemSizeChoices **Integer - Read/Write**

Returns the number of memsize choices that the current processor type has. If you look at the controller properties of a project, you will see where you can select the platform, processor, and series, and there is a list box for selecting a memory size. This property will tell you how many choices you have to select from. Most times it is only one.



Online **Boolean - Read Only**

Returns whether or not the processor is online.



OnlineChangesMade **Boolean - Read Only**

Queries and returns whether or not any online changes have been made.








ProcessorMode **IgxProcOnlineState - Read/Write**

Returns or sets the current mode of the processor. This can be set to one of the following when examining this property:

- (6) IgxRemoteProg
- (7) IgxRemoteTest
- (8) IgxRemoteRun

See Appendix B for a complete list of type definitions for `IgxProcOnlineState`.

<i>(500 only)</i>		ProgramID	Integer - Read Only
<hr/>			
Returns the 4-byte error check (CRC) of the program.			
<i>(5 only)</i>		Revision	Integer - Read/Write
<hr/>			
Sets or returns the revision number of the processor			
<i>(5 only)</i>		Series	Integer - Read/Write
<hr/>			
Sets or returns the series of the processor			
<i>(5 only)</i>		Subrevision	Integer - Read/Write
<hr/>			
Sets or returns the subrevision of the processor.			
		Type	IgxProcessorTypeConstant - Read Only
<hr/>			
Returns the type of the processor as a <code>IgxProcessorTypeConstant</code> .			

Methods

Using a method causes something to happen to an object. In most cases methods are actions. Use any of the following methods to identify an action for the Processor object to perform. Although written for the RSLogix 5 software product, the short examples following each method may be easily adapted to RSLogix 500. For example, type definitions may vary between products, and those differences must be considered when adapting code to the RSLogix 500 object model.

ClearAllForces

Use this method to remove all forces from the input and output force tables.

Syntax

```
ClearAllForces()
```

Example

The following code snippet clears all forces in the current processor.

```
Call gProc.ClearAllForces()
```



DisableForces

Use this method to disable all forced I/O bits.

Syntax

```
DisableForces()
```

Example

The following code snippet disables all forces in the current processor.

```
Call gProc.DisableForces()
```



EnableForces

Use this method to enable all forced I/O bits. Enabling the input force table affects the input force table, input data file, and also the program logic. Enabling the output force table only affects the output circuit; it does not affect the output data file or program logic. Use caution when enabling forces.

Syntax

```
EnableForces()
```

Example

The following code snippet enables all forces in the current processor.

```
Call gProc.EnableForces()
```

(5 only)



GetPLC5MemSizeChoiceByIndex Long

Use this method to get any of the legal memory size choices for the currently selected processor.

Syntax

```
GetPLC5MemSizeChoiceByIndex(Index as Short) as Long
```

Arguments

Index - If property NumberOfMemSizeChoices returns 2, then *Index* can equal either 1 or 2 in order to return the desired Memory size. Most times NumberOfMemSizeChoices will return 1, so *Index* will be 1. You can use an integer for this parameter in Visual Basic.

Example

The following code snippet returns the memory size.

```
Dim MemSize As Long
```

```
MemSize = gProcessor.GetPLC5MemSizeChoiceByIndex(1)
```

(5 only)



SetPLC5MemSize

Boolean

Use this method to set the memory size of the processor.

Syntax

SetPLC5MemSize (*MemSize as Long*) as Boolean

Arguments

MemSize - You can get valid memory sizes for the selected processor by using `NumberOfMemSizeChoices` and `GetPLC5MemSizeChoiceByIndex`.

Returns

Returns `True` if successful, otherwise `false` is returned. `False` would be returned if the `nMemSize` did not match any of the legal memory sizes acquired by `GetPLC5MemSizeChoiceByIndex`.

Example

The following code snippet sets the memory size of the current processor.

```
Dim Result As Boolean
Dim MemSize As Long
MemSize = gProcessor.GetPLC5MemSizeChoiceByIndex(1)
Result = gProcessor.SetPLC5MemSize (MemSize)
```

Events

There are no events defined for the Processor object.

Summary example

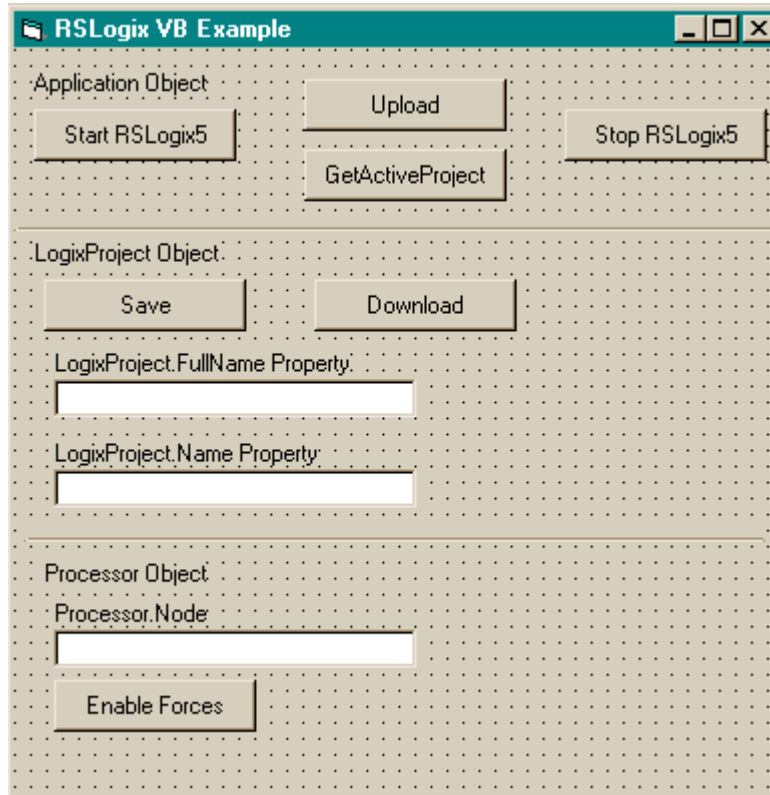
Important

This book assumes that you have the basic knowledge required to work with forms and controls in Visual Basic.

The following example automates functionality within RSLogix 5 with the automation interface by incorporating properties, methods and events from the Application, LogixProject and Processor objects. Comments within the code are preceded by an apostrophe ('). You'll see that although the example is specific to RSLogix 5 software, it is generic enough to adapt to RSLogix 500 with only minor form and comment alterations.

Form

The following form builds on the forms first presented in Chapters 2 and 3. Subsequent chapters in this book will continue to build on this form as new objects are introduced.



The screenshot shows a Windows-style application window titled "RSLogix VB Example". The interface is organized into three main sections, each with a dotted background:

- Application Object:** Contains five buttons: "Start RSLogix5", "Upload", "GetActiveProject", and "Stop RSLogix5".
- LogixProject Object:** Contains two buttons: "Save" and "Download". Below the buttons are two text input fields labeled "LogixProject.FullName Property:" and "LogixProject.Name Property:".
- Processor Object:** Contains one text input field labeled "Processor.Node:" and one button labeled "Enable Forces".

Code

```
'-----  
' Global variables  
'-----  
  
Dim gApplication As RSLogix5.Application 'Application object  
Dim gProject As RSLogix5.LogixProject 'LogixProject object  
Dim gProcessor As RSLogix5.Processor 'Processor object  
  
'-----  
' Application  
'-----  
  
Private Sub Command1_Click()  
  
    ' Set application object to the object returned from CreateObject.  
    ' CreateObject is simply a method provided by Microsoft that creates  
    ' a new registered COM application instance. In this case we start  
    ' RSLogix 5 by using the "RSLogix5.Application" string.  
    Set gApplication = CreateObject("RSLogix5.Application")  
    ' At this point, if the CreateObject method functioned properly, the  
    ' gApplication object is now a direct reference to the RSLogix5  
    ' Object Model. Any properties or methods that we invoke on this  
    ' object will immediately take effect in RSLogix.  
  
    ' Immediately set the visible property of the application to 'True'  
    gApplication.Visible = True  
  
    ' Assign the AutoSaveInterval value to 3 minutes  
    gApplication.AutoSaveInterval = 3  
  
    ' Assign WindowState prop to lgxWindowStateMaximized enumeration  
    gApplication.WindowState = lgxWindowStateMaximized  
  
End Sub  
  
Private Sub Command2_Click()  
    ' Quit the application ignoring prompts and not saving changes.  
    gApplication.Quit True, False  
    ' Eliminate the reference to the application object  
    Set gApplication = Nothing  
End Sub  
  
Private Sub Command3_Click()  
On Error GoTo errorHandler  
    ' Upload a project from the processor using the upload method of the  
    ' application object while ignoring prompts, NOT saving the previous  
    ' file, creating new file from the upload (using lgxUploadCreateNew  
    ' enum (see the objectbrowser for more enumerations)), and go online  
    ' (using the lgxGoOnline enum).
```

```

        ' Set the returned object reference to the gProject object.
        Set gProject = gApplication.Upload(True, False, lgxUploadCreateNew,
lgxGoOnline)
Exit Sub

errorHandler:
    ' Upon a caught error decide what to do.
    MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Command4_Click()
    ' Get the currently open project in the application.
    Set gProject = gApplication.GetActiveProject
End Sub

'-----
' LogixProject
'-----

Private Sub Command5_Click()
    Dim returnValue As Boolean

    ' Save the currently open project, assign the return value to a
    ' variable and display that value in a message box.
    returnValue = gProject.Save(True, True)
    MsgBox "Returned: " & returnValue

End Sub

Private Sub Command6_Click()
On Error GoTo errorHandler
    Dim returnValue As Boolean

    ' Download the project to the current processor. This method call is
    ' ignoring all prompts, going online (using the lgxGoOnline enum),
    ' setting the processor to remote program mode (using the
    ' lgxREMOTEPROG enum) and displaying the return value in a message
    ' box
    returnValue = gProject.Download(True, lgxGoOnline, lgxREMOTEPROG)
    MsgBox "Returned: " & returnValue
Exit Sub

errorHandler:
    ' Upon a caught error decide what to do.
    MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Text1_Click()
    ' Display the current name of the project in a text box.

```

```

        Text1.Text = gProject.FullName
        Text2.Text = gProject.Name
    End Sub

'-----
' Processor
'-----

Private Sub Text3_Click()
    ' Set the processors reference to a global variable.
    Set gProcessor = gProject.Processor

    ' Display the current node in a text box.
    Text3.Text = gProcessor.Node
End Sub

Private Sub Command7_Click()
    ' Set the processors reference to a global variable.
    Set gProcessor = gProject.Processor

    ' Enable forces in the processor
    gProcessor.EnableForces
End Sub

```


Chapter
5

ProgramFiles collection

The ProgramFiles collection represents all the program files in the project. The ProgramFiles collection can be obtained from the “ProgramFiles” property of the LogixProject object. The ProgramFiles collection is not creatable with the CreateObject function.

Properties	Methods	Events
Application	Add Count Item Remove	-None-

The following commented code example illustrates how you might get the ProgramFiles collection from the LogixProject object. The example adds error checking and displays a message if the RSLogix application can find no program files.

```
'get the programfiles object
Set gProgFiles = gProject.ProgramFiles
If gProgFiles Is Nothing Then
  'if the programfiles object does not exist then display an error
  MsgBox "RSLogix could not get Program Files!", vbExclamation,
  "ERROR: 008"
  Exit Function
End If
```

Properties

In most cases properties are characteristics or attributes of an object. Using a property returns information about the object or causes a quality of the object to change. The following properties define the ProgramFiles collection.



Application

Application - Read Only

This property returns an Application object that represents the RSLogix application.

Methods

Using a method causes something to happen to an object. In most cases methods are actions. Use any of the following methods to identify an action for the ProgramFiles collection to perform. Although written for the RSLogix 5 software product, the short examples following each method may be easily adapted to RSLogix 500. For example, type definitions may vary between products, and those differences must be considered when adapting code to the RSLogix 500 object model.



Add

ProgramFile

Use this method to create a new program file and add it to the ProgramFile collection.

Syntax

Add(*FileNumber* as Integer, *FileType* as lgxProgramFileTypeConstants, *Debug* as Boolean, *IgnorePrompts* as Boolean) as ProgramFile

Arguments

FileNumber - The number for the program file to be created.

FileType - Choose from the following lgxProgramFileTypeConstants. See also Appendix B for descriptions of the possible selections.

- (1) lgxLADDER
- (2) lgxSFCNEW
- (3) lgxSFCOLD
- (4) lgxSTX
- (9) lgxCAR

Debug - Select True to make the file a debug file, otherwise False.

IgnorePrompts - If set to True no user interface prompts are displayed to the user. If False prompts are displayed.


Returns

If successful a new program file (defined by the supplied parameters) is created and added to the Program Files collection and a reference to the new program file is returned. If unsuccessful Nothing is returned.

Example

The following code snippet makes the call to RSLogix 5 to add program file #7 (a ladder logic file) to the program files collection. This file will not be a debug file, and no user prompts will inform the user of its creation.

```
Set gProgramFile = gProgramFiles.Add(7, lgxLADDER, False, True)
```

	Count	Long
---	-------	------

Use this method to return the number of program file objects in the ProgramFiles collection.

Syntax

Count() As Long

Returns

If successful the number of program file objects in the collection is returned. This includes any unused program files between the first and last files defined in the project.

Example

The following code snippet displays the number of program files in your project.

```
MsgBox "Number of Program Files = " & gProgramFiles.Count
```

	Item	ProgramFile
---	------	-------------

Use this method to retrieve a specified program file from the ProgramFiles collection.

Syntax

Item(*Index as Long*) As ProgramFile

Arguments

Index - The value of index should be between 0 and Count-1 inclusive. This represents the file number to be retrieved.

Returns

If successful the program file (specified by the index) is returned; otherwise returns Nothing.

Example

The following code snippet displays the name of a specific program file returned by the Item method.

```
Text1.text = gProgramFiles.Item(FileNumber).Name
```



Remove**Boolean**

Use this method to remove a program file from the ProgramFiles collection.

Syntax

Remove(*FileNumber as Integer, IgnorePrompts as Boolean*) as Boolean

Arguments

FileNumber - The number of the program file you want removed.

IgnorePrompts - If set to True no user interface prompts are displayed to the user. If False prompts are displayed.

Returns

If successful the designated program file is removed from the Program Files collection and a value of True is returned; if unsuccessful False is returned.

Example

The following code snippet makes the call to remove program file #7 from the program files collection. No user prompts will inform the user of its removal.

```
Result = gProgramFiles.Remove(7, True)
```

Events

No events have been defined for the ProgramFiles object/collection.

Summary example

Important

This book assumes that you have the basic knowledge required to work with forms and controls in Visual Basic.

The following example automates functionality within RSLogix 5 with the automation interface by incorporating properties, methods and events from the Application, LogixProject and Processor objects as well as the ProgramFiles collection. Comments within the code are preceded by an apostrophe ('). You'll see that although the example is specific to RSLogix 5 software, it is generic enough to adapt to RSLogix 500 with only minor form and comment alterations.

Form

The following form builds on the forms first presented in Chapters 2, 3 and 4. Subsequent chapters in this book will continue to build on this form as new objects are introduced.

The screenshot shows a Windows-style application window titled "RSLogix VB Example". The window has a standard title bar with minimize, maximize, and close buttons. The main area is divided into four sections, each with a dotted background:

- Application Object:** Contains four buttons: "Start RSLogix5", "Upload", "GetActiveProject", and "Stop RSLogix5".
- LogixProject Object:** Contains two buttons: "Save" and "Download". Below them are two text input fields labeled "LogixProject.FullName Property" and "LogixProject.Name Property".
- Processor Object:** Contains one text input field labeled "Processor.Node" and one button labeled "Enable Forces".
- Program Files Collection:** Contains one text input field labeled "Program File:" and two buttons labeled "Add" and "Remove".

Code

```
'-----  
' Global variables  
'-----  
  
Dim gApplication As RSLogix5.Application 'Application object  
Dim gProject As RSLogix5.LogixProject 'LogixProject object  
Dim gProcessor As RSLogix5.Processor 'Processor object  
Dim gProgramFiles As RSLogix5.ProgramFiles 'ProgramFiles object  
Dim gProgramFile As RSLogix5.ProgramFile 'ProgramFile object
```

```

'-----
' Application
'-----

Private Sub Command1_Click()

    ' Set application object to the object returned from CreateObject.
    ' CreateObject is simply a method provided by Microsoft that creates
    ' a new registered COM application instance. In this case we start
    ' RSLogix 5 by using the "RSLogix5.Application" string.
    Set gApplication = CreateObject("RSLogix5.Application")
    ' At this point, if the CreateObject method functioned properly, the
    ' gApplication object is now a direct reference to the RSLogix5
    ' Object Model. Any properties or methods that we invoke on this
    ' object will immediately take effect in RSLogix.

    ' Immediately set the visible property of the application to 'True'
    gApplication.Visible = True

    ' Assign the AutoSaveInterval value to 3 minutes
    gApplication.AutoSaveInterval = 3

    ' Assign WindowState prop to lgxWindowStateMaximized enumeration
    gApplication.WindowState = lgxWindowStateMaximized

End Sub

Private Sub Command2_Click()
    ' Quit the application ignoring prompts and not saving changes.
    gApplication.Quit True, False
    ' Eliminate the reference to the application object
    Set gApplication = Nothing
End Sub

Private Sub Command3_Click()
On Error GoTo errorHandler
    ' Upload a project from the processor using the upload method of the
    ' application object while ignoring prompts, NOT saving the previous
    ' file, creating new file from the upload (using lgxUploadCreateNew
    ' enum (see the objectbrowser for more enumerations)), and go online
    ' (using the lgxGoOnline enum).
    ' Set the returned object reference to the gProject object.
    Set gProject = gApplication.Upload(True, False, lgxUploadCreateNew,
    lgxGoOnline)
Exit Sub

errorHandler:
    ' Upon a caught error decide what to do.
    MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
    Err.Description
End Sub

```

```

Private Sub Command4_Click()
    ' Get the currently open project in the application.
    Set gProject = gApplication.GetActiveProject
End Sub

'-----
' LogixProject
'-----

Private Sub Command5_Click()
    Dim ReturnValue As Boolean

    ' Save the currently open project, assign the return value to a
    ' variable and display that value in a message box.
    ReturnValue = gProject.Save(True, True)
    MsgBox "Returned: " & ReturnValue

End Sub

Private Sub Command6_Click()
On Error GoTo errorHandler
    Dim ReturnValue As Boolean

    ' Download the project to the current processor. This method call is
    ' ignoring all prompts, going online (using the lgxGoOnline enum),
    ' setting the processor to remote program mode (using the
    ' lgxREMOTEPROG enum) and displaying the return value in a message
    ' box
    ReturnValue = gProject.Download(True, lgxGoOnline, lgxREMOTEPROG)
    MsgBox "Returned: " & ReturnValue
Exit Sub

errorHandler:
    ' Upon a caught error decide what to do.
    MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Text1_Click()
    ' Display the current name of the project in a text box.
    Text1.Text = gProject.FullName
    Text2.Text = gProject.Name
End Sub

'-----
' Processor
'-----

Private Sub Text3_Click()
    ' Set the processors reference to a global variable.
    Set gProcessor = gProject.Processor

```

```

    ' Display the current node in a text box.
    Text3.Text = gProcessor.Node
End Sub

Private Sub Command7_Click()
    ' Set the processors reference to a global variable.
    Set gProcessor = gProject.Processor

    ' Enable forces in the processor
    gProcessor.EnableForces
End Sub

'-----
' Program Files Collection
'-----

Private Sub Command8_Click()
    ' Set current programfiles collection reference from the project to
    ' a global variable.
    Set gProgramFiles = gProject.ProgramFiles

    ' Add new ladder file into the ProgramFiles collection. This method
    ' call sets the gProgramFile object to a new ProgramFile object
    ' created at the file number specified by the value of text4.text,
    ' using the lgxLADDER enum to specify to create a ladder file,
    ' that is NOT a debug file, and ignoring all prompts.
    Set gProgramFile = gProgramFiles.Add(CInt(Text4.Text), lgxLADDER,
    False, True)

End Sub

Private Sub Command9_Click()
    Dim ReturnValue As Boolean

    ' Set current programfiles collection reference from the project to
    ' a global variable.
    Set gProgramFiles = gProject.ProgramFiles

    ' Remove the ProgramFile specified in the ext box from the
    ' ProgramFiles collection.
    ' Display the returned value in a message box.
    ReturnValue = gProgramFiles.Remove(CInt(Text4.Text), True)
    MsgBox "Returned: " & ReturnValue

End Sub

```


Chapter
6

ProgramFile object

The ProgramFile object represents the base functionality of a program file. It is obtained when using the Item or Add methods.

You cannot create a separate instance of the ProgramFile object with the CreateObject function.

Properties	Methods	Events
Application	-None-	-None-
Debug		
DefaultName		
Description		
FileNumber		
FormattedName		
InUse		
MaxDescriptionLength		
MaxNameLength		
Name		
Online		
Programmable		
ProtectionSupported - (RSLogix 500 only)		
ReadPrivilege - (RSLogix 5 only)		
Reserved - (RSLogix 500 only)		
Type		
WritePrivilege - (RSLogix 5 only)		


The following commented code example illustrates a typical call to the main program file (ladder file #2).

```
Global gApplication As RSLogix5.Application


Sub buttonLoadFile2_Click()
    Dim ProgramFiles As RSLogix5.ProgramFiles
    Dim ProgramFile As RSLogix5.Programfile
    Dim CurrentProject As RSLogix5.LogixProject
    On Error Resume Next
    Set CurrentProject = gApplication.GetActiveProject()
    If Not CurrentProject Is Nothing Then
        Set ProgramFiles = CurrentProject.ProgramFiles
        Set Programfile = ProgramFiles(2)
        If Not Programfile Is Nothing Then
            'Okay to use ProgramFile object...
        End If
    End If
End Sub
```

Properties

In most cases properties are characteristics or attributes of an object. Using a property returns information about the object or causes a quality of the object to change. The following properties define the ProgramFile object.

 Application	Application - Read Only
--	--------------------------------


This property returns the Application object.

 Debug	Boolean - Read Only
--	----------------------------


This property returns whether or not the program file is for debug use only.

 DefaultName	String - Read Only
--	---------------------------











This property returns the default name of the file.

 Description	String - Read/Write
--	----------------------------

This property returns or sets the description of the program file.

 FileNumber	Long - Read Only
---	-------------------------

This property returns the file number.

	 FormattedName	String - Read Only
	This property returns formatted name of the file. The format returned is as follows: SYS 0, LAD 2, SFC 4, or STX 6.	
	 InUse	Boolean - Read Only
	This property returns whether or not the file is being used.	
	 MaxDescriptionLength	Long - Read Only
	This property returns the maximum allowable characters for the file description.	
	 MaxNameLength	Long - Read Only
	This property returns the maximum allowable characters for the file name.	
	 Name	String - Read/Write
	This property returns or sets the name of the file.	
	 Online	Boolean - Read Only
	This property returns whether or not the project controlling this program file is currently online with the processor.	
	 Programmable	Boolean - Read Only
	This property returns whether or not the program file is programmable.	
<i>(500 only)</i>	 ProtectionSupported	Boolean - Read Only
	This property returns the attribute of protection supported by this program file.	
<i>(5 only)</i>	 ReadPrivilege	Boolean - Read Only
	This property returns whether or not under the current privilege class the program file is read-enabled. This is a feature available only to processors with the passwords and privileges functionality.	
<i>(500 only)</i>	 Reserved	Boolean - Read Only
	This property returns True if the program file is reserved.	

**Type****IgxProgramFileTypeConstants - Read Only**

This property returns the type of file. Possible returned types are listed below and described in Appendix B. .

- (0) IgxHEADER
- (1) IgxLADDER
- (2) IgxSFCNEW
- (3) IgxSFCOLD
- (4) IgxSTX
- (9) IgxCAR

(5 only)**WritePrivilege****Boolean - Read Only**

This property returns whether or not under the current privilege class the program file is write-enabled. This is a feature available only to processors with the passwords and privileges functionality.

Methods

There are no methods defined for the ProgramFile object.

Events

There are no events defined for the ProgramFile object.

Summary example

Important

This book assumes that you have the basic knowledge required to work with forms and controls in Visual Basic.

The following example automates functionality within RSLogix 5 with the automation interface by incorporating properties, methods and events from the Application, LogixProject and Processor objects, the ProgramFiles collection and the ProgramFile object. Comments within the code are preceded by an apostrophe ('). You'll see that although the example is specific to RSLogix 5 software, it is generic enough to adapt to RSLogix 500 with only minor form and comment alterations.

Form

The following form builds on the forms first presented in Chapters 2, 3, 4 and 5. Subsequent chapters in this book will continue to build on this form as new objects are introduced.

The screenshot shows a Windows-style application window titled "RSLogix VB Example". The window contains several sections of controls on a dotted background:

- Application Object:** Contains buttons for "Start RSLogix5", "Upload", "GetActiveProject", and "Stop RSLogix5".
- LogixProject Object:** Contains buttons for "Save" and "Download". Below these are two text input fields labeled "LogixProject.FullName Property:" and "LogixProject.Name Property:".
- Processor Object:** Contains a text input field labeled "Processor.Node:" and a button labeled "Enable Forces".
- Program Files Collection:** Contains a text input field labeled "Program File:", followed by buttons for "Add", "Remove", and "Get", and a label "For ProgramFile.".
- ProgramFile:** Contains two text input fields labeled "Type:" and "Name:".

Code

```
'-----
' Global variables
'-----

Dim gApplication As RSLogix5.Application 'Application object
Dim gProject As RSLogix5.LogixProject 'LogixProject object
Dim gProcessor As RSLogix5.Processor 'Processor object
Dim gProgramFiles As RSLogix5.ProgramFiles 'ProgramFiles object
Dim gProgramFile As RSLogix5.ProgramFile 'ProgramFile object

'-----
' Application
'-----

Private Sub Command1_Click()

    ' Set application object to the object returned from CreateObject.
    ' CreateObject is simply a method provided by Microsoft that creates
    ' a new registered COM application instance. In this case we start
    ' RSLogix 5 by using the "RSLogix5.Application" string.
    Set gApplication = CreateObject("RSLogix5.Application")
    ' At this point, if the CreateObject method functioned properly, the
    ' gApplication object is now a direct reference to the RSLogix5
    ' Object Model. Any properties or methods that we invoke on this
    ' object will immediately take effect in RSLogix.

    ' Immediately set the visible property of the application to 'True'
    gApplication.Visible = True

    ' Assign the AutoSaveInterval value to 3 minutes
    gApplication.AutoSaveInterval = 3

    ' Assign WindowState prop to lgxWindowStateMaximized enumeration
    gApplication.WindowState = lgxWindowStateMaximized

End Sub

Private Sub Command2_Click()
    ' Quit the application ignoring prompts and not saving changes.
    gApplication.Quit True, False
    ' Eliminate the reference to the application object
    Set gApplication = Nothing
End Sub

Private Sub Command3_Click()
    On Error GoTo errorHandler
    ' Upload a project from the processor using the upload method of the
```

```

    ' application object while ignoring prompts, NOT saving the previous
    ' file, creating new file from the upload (using lgxUploadCreateNew
    ' enum (see the objectbrowser for more enumerations)), and go online
    ' (using the lgxGoOnline enum).
    ' Set the returned object reference to the gProject object.
    Set gProject = gApplication.Upload(True, False, lgxUploadCreateNew,
lgxGoOnline)
Exit Sub

errorHandler:
    ' Upon a caught error decide what to do.
    MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Command4_Click()
    ' Get the currently open project in the application.
    Set gProject = gApplication.GetActiveProject
End Sub

'-----
' LogixProject
'-----

Private Sub Command5_Click()
    Dim ReturnValue As Boolean

    ' Save the currently open project, assign the return value to a
    ' variable and display that value in a message box.
    ReturnValue = gProject.Save(True, True)
    MsgBox "Returned: " & ReturnValue

End Sub

Private Sub Command6_Click()
On Error GoTo errorHandler
    Dim ReturnValue As Boolean

    ' Download the project to the current processor. This method call is
    ' ignoring all prompts, going online (using the lgxGoOnline enum),
    ' setting the processor to remote program mode (using the
    ' lgxREMOTEPROG enum) and displaying the return value in a message
    ' box
    ReturnValue = gProject.Download(True, lgxGoOnline, lgxREMOTEPROG)
    MsgBox "Returned: " & ReturnValue
Exit Sub

errorHandler:
    ' Upon a caught error decide what to do.
    MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description

```

```

End Sub

Private Sub Text1_Click()
    ' Display the current name of the project in a text box.
    Text1.Text = gProject.FullName
    Text2.Text = gProject.Name
End Sub

'-----
' Processor
'-----

Private Sub Text3_Click()
    ' Set the processor's reference to a global variable.
    Set gProcessor = gProject.Processor

    ' Display the current node in a text box.
    Text3.Text = gProcessor.Node
End Sub

Private Sub Command7_Click()
    ' Set the processor's reference to a global variable.
    Set gProcessor = gProject.Processor

    ' Enable forces in the processor
    gProcessor.EnableForces
End Sub

'-----
' Program Files Collection
'-----

Private Sub Command8_Click()
    ' Set current programfiles collection reference from the project to
    ' a global variable.
    Set gProgramFiles = gProject.ProgramFiles

    ' Add new ladder file into the ProgramFiles collection. This method
    ' call sets the gProgramFile object to a new ProgramFile object
    ' created at the file number specified by the value of text4.text,
    ' using the lgxLADDER enum to specify to create a ladder file,
    ' that is NOT a debug file, and ignoring all prompts.
    Set gProgramFile = gProgramFiles.Add(CInt(Text4.Text), lgxLADDER,
    False, True)

End Sub

Private Sub Command9_Click()
    Dim ReturnValue As Boolean

    ' Set the current programfiles collection reference from the project

```



```

' to a global variable.
Set gProgramFiles = gProject.ProgramFiles

' Remove the ProgramFile specified in the ext box from the
' ProgramFiles collection.
' Display the returned value in a message box.
ReturnValue = gProgramFiles.Remove(CInt(Text4.Text), True)
MsgBox "Returned: " & ReturnValue

End Sub

'-----
' ProgramFile Object
'-----

Private Sub Command10_Click()

' Set the current programfiles collection reference from the project
' to a global variable.
Set gProgramFiles = gProject.ProgramFiles

' Set the programfile reference specified by the value of a textbox
' to the current global object.
Set gProgramFile = gProgramFiles(CInt(Text4.Text))

End Sub

Private Sub Text5_Click()
' Assign the textboxes the Type and Name of the programfile.
Text5.Text = gProgramFile.Type
Text6.Text = gProgramFile.Name

End Sub

Private Sub Text6_Change()
' Change the ProgramFile.Name property to the current text.
gProgramFile.Name = Text6.Text

End Sub

```


Chapter
7

DataFiles collection

The DataFiles collection represents the collection of data files in the RSLogix project. The DataFiles collection can be obtained using the DataFiles property of the LogixProject object. The DataFiles collection is not creatable with the CreateObject function.

Properties	Methods	Events
Application	Add Count GetDataValue Item Remove SetDataValue	-None-

The following commented code example illustrates how you might get the DataFiles collection from the LogixProject object. This example adds error checking and notification.

```
'get the DataFiles collection from the LogixProject object
Set gDataFiles = gLogixProject.DataFiles
'if Logix failed to get the Data Files collection then display an
'error and exit
If gDataFiles Is Nothing Then
    MsgBox "ERROR: Could not get Data Files!", vbExclamation, "ERROR"
    Exit Function
End If
```

Properties

In most cases properties are characteristics or attributes of an object/collection. Using a property returns information or causes a quality of the object/collection to change. The following properties define the DataFiles collection.



Application**Application - Read Only**

This property returns an Application object that represents the RSLogix 5 or 500 application.

Methods

Using a method causes something to happen to an object. In most cases methods are actions. Use any of the following methods to identify an action for the DataFiles object (collection) to perform. Although written for RSLogix 5, the short examples following each method may be easily adapted to RSLogix 500. For example, type definitions may vary between products, and those differences must be considered when adapting code to RSLogix 500.



Add**DataFile**

Use this method to create a new data file and add it to the DataFiles collection.

Syntax

Add(FileName as Integer, FileType as lgxDataFileTypeConstants, NumberOfElements as Integer, IgnorePrompts as Boolean) as DataFile

Arguments

FileName - The number of the data file to add.

FileType - The type of data file to add. The valid list is provided and defined in Appendix B.

NumberOfElements - The number of elements in the data file to add.

IgnorePrompts - When True no user interface confirmations will be displayed. If FALSE prompts are displayed.


Returns

If successful the data file object is created and added to the DataFiles collection and a reference to the newly created data file is returned. If unsuccessful Nothing is returned.

Example

The following code snippet makes the call to RSLogix 5 to add a binary file #20 to the data files collection. This file will have 45 elements, and no user prompts will inform the user of its creation.

```
Set gDataFile = gDataFiles.Add(20, lgxDTBINARY, 45, True)
```

 Count	Long
--	-------------

Use this method to return the number of data file objects in the collection.

Syntax

Count() As Long


Returns

If successful the number of data file objects in the collection is returned.

Example

The following code snippet displays the number of data files in your project.

```
MsgBox "Number of Data Files = " & gDataFiles.Count
```

 GetDataValue	String
---	---------------

Use this method to return the current data value of a specified data address.

Syntax

GetDataValue(*Address As String*) As String

Arguments

Address - The string address for the data to be read.

Returns

If successful the current data value for the address that you specify is returned as a string.

Example

The following code snippet returns the value of the accumulator in Timer T4:0.

```
Dim value as String  
value = gDataFiles.GetDataValue("T4:0.acc")
```

 Item	Data File
---	------------------

Use this method to retrieve a data file.

Syntax

Item(*Index as Long*) As DataFile

Arguments

Index - The value of index should be between 0 and Count-1 inclusive. This represents the number of the data file to be retrieved.

Returns

If successful the data file object (specified by the index) is returned. If unsuccessful Nothing is returned.

Example

The following code snippet displays the name of a data file retrieved by the Item property in a text box.

```
text1.Text = gDataFiles.Item(Data_File).Name
```



Remove

Boolean

Use this method to remove a data file from the DataFiles collection.

Syntax

Remove(*FileNumber as Integer, IgnorePrompts as Boolean*) As Boolean

Arguments

FileNumber - The number of the file to remove.

IgnorePrompts - When True no user interface confirmations will be displayed. If FALSE prompts are displayed.

Returns

If successful the indicated data file is removed from the DataFiles collection and a value of True is returned; if unsuccessful False is returned.

Example

The following code snippet removes data file #11 from the project without issuing any prompts first.

```
Result = gDataFiles.Remove(11, True)
```



SetDataValue

Boolean

Use this method to write a data value to a data address.

Syntax

SetDataValue(*Address as String, Value as String*) as Boolean

Arguments

Address - The string address to be written to.

Value - The value to be written to the data file.

Returns

If successful the value is written and True is returned; if unsuccessful False is returned.

Example

The following code snippet sets the value of the T4:0 timer preset to 60.

```
Res = gDataFiles.SetDataValue("T4:0.pre", "60")
```

Events

There are no events defined for the DataFiles object.

Summary Example

Important

This book assumes that you have the basic knowledge required to work with forms and controls in Visual Basic.

The following example automates functionality within RSLogix 5 with the automation interface by incorporating properties, methods and events from the Application and LogixProject object. Comments within the code are preceded by an apostrophe ('). You'll see that although the example is specific to RSLogix 5 software, it is generic enough to adapt to RSLogix 500 with only minor form and comment alterations.

Form

The following form builds on the forms first presented in Chapters 2 and 3.

The screenshot shows a Windows-style application window titled "RSLogix VB Example". The window contains several sections of controls:

- Application Object:** Contains three buttons: "Start RSLogix5", "Upload", and "Stop RSLogix5".
- LogixProject Object:** Contains two buttons: "Save" and "Download".
- LogixProject.FullName Property:** A text input field.
- LogixProject.Name Property:** A text input field.
- DataFiles Collection:** Contains a "File Number" label and a text input field, followed by a "Get DataFile" button.
- Address:** A text input field.
- Value:** A text input field.
- SetData:** A button.

Code

```
'-----  
' Global variables  
'-----  
  
Dim gApplication As RSLogix5.Application 'Application object  
Dim gProject As RSLogix5.LogixProject 'LogixProject object  
Dim gDataFiles As RSLogix5.DataFiles 'DataFiles Collection  
Dim gDataFile As RSLogix5.DataFile 'DataFile object  
  
'-----  
' Application  
'-----  
  
Private Sub Command1_Click()  
  
    ' Set the application object to object returned from CreateObject.  
    ' CreateObject is simply a method provided by Microsoft that creates  
    ' a new registered COM application instance. In this case we start  
    ' RSLogix 5 by using the "RSLogix5.Application" string.  
    Set gApplication = CreateObject("RSLogix5.Application")  
    ' At this point, if the CreateObject method functioned properly, the  
    ' gApplication object is now a direct reference to the RSLogix5  
    ' Object Model. Any properties or methods that we invoke on this  
    ' object will immediately take effect in RSLogix.  
  
    ' Immediately set the visible property of the application to 'True'  
    gApplication.Visible = True  
  
    ' Assign the AutoSaveInterval value to 3 minutes  
    gApplication.AutoSaveInterval = 3  
  
    ' Assign WindowState prop to lgxWindowStateMaximized enumeration  
    gApplication.WindowState = lgxWindowStateMaximized  
  
End Sub  
  
Private Sub Command2_Click()  
    ' Quit the application ignoring prompts and not saving changes.  
    gApplication.Quit True, False  
    ' Eliminate the reference to the application object  
    Set gApplication = Nothing  
End Sub  
  
Private Sub Command3_Click()  
On Error GoTo errorHandler  
    ' Upload a project from the processor using the upload method of the  
    ' application object while ignoring prompts, NOT saving the previous  
    ' file, creating new file from the upload (using lgxUploadCreateNew  
    ' enum (see objectbrowser for more enumerations)), and going online
```

```

' (using the lgxGoOnline enum).
' Set the returned object reference to the gProject object.
Set gProject = gApplication.Upload(True, False, lgxUploadCreateNew,
lgxGoOnline)
Exit Sub

errorHandler:
' Upon a caught error decide what to do.
MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Command4_Click()
' Get the currently open project in the application.
Set gProject = gApplication.GetActiveProject
End Sub

'-----
' LogixProject
'-----

Private Sub Command5_Click()
Dim ReturnValue As Boolean

' Save the currently open project, assign return value to a variable
' and display that value in a message box.
ReturnValue = gProject.Save(True, True)
MsgBox "Returned: " & ReturnValue

End Sub

Private Sub Command6_Click()
On Error GoTo errorHandler
Dim ReturnValue As Boolean

' Download the project to the current processor. This method call is
' ignoring all prompts, going online (using the lgxGoOnline enum),
' setting the processor to remote program mode (using the
' lgxREMOTEPROG enum) and displaying the return value in a message
' box
ReturnValue = gProject.Download(True, lgxGoOnline, lgxREMOTEPROG)
MsgBox "Returned: " & ReturnValue
Exit Sub

errorHandler:
' Upon a caught error decide what to do.
MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Text1_Click()

```

```

        ' Display the current name of the project in a text box.
        Text1.Text = gProject.FullName
        Text2.Text = gProject.Name
    End Sub

    '-----
    ' DataFiles
    '-----

    Private Sub Command7_Click()
        ' Set the current datafiles reference to a global variable.
        Set gDataFiles = gProject.DataFiles

        ' Retrieve the datafile reference for file number entered
        ' by the user in the text box.
        Set gDataFile = gDataFiles.Item(CInt(Text3.Text))

    End Sub

    Private Sub Command8_Click()
        Dim ReturnValue As Boolean

        ' Set the current datafiles reference to a global variable.
        Set gDataFiles = gProject.DataFiles

        ' Set a data value at a user specified address to a user specified
        ' value
        ReturnValue = gDataFiles.SetDataValue(Text4.Text, Text5.Text)

    End Sub

```


Chapter
8

DataFile object

The DataFile object represents a data file in the project or processor. The DataFile object is obtained from the DataFiles collection via the Add and Item methods. You cannot create a new instance of a DataFile object with the CreateObject function.








Properties	Methods	Events
Application	-None-	-None-
CanBeDeleted		
CanBeMonitored		
CanChangeScope		
CanChangeSize		
Debug		
Description		
FileNumber		
FormattedName		
GlobalScope		
InUse		
LocalScope		
MaxDescriptionLength		
MaxNameLength		
Name		
NumberOfElements		
Online		
ReadPrivilege (<i>RSLogix 5 only</i>)		
Reserved (<i>RSLogix 500 only</i>)		
Scopeable		
Type		
TypeAsString		
WritePrivilege (<i>RSLogix 5 only</i>)		











The following commented code example illustrates how you might access the DataFile object.







```
Private Sub Form_Load()  
Set gDataFiles = gProject.DataFiles  
Set gDataFile = gDataFile(6)  
If gDataFile Is Nothing Then  
'if the DataFile object does not exist then display an error  
MsgBox "Error getting Data File"  
End If
```

Properties

In most cases properties are characteristics or attributes of an object. Using a property returns information about the object or causes a quality of the object to change. The following properties define the DataFile object.

 Application	Application - Read Only
Returns an Application object that represents the RSLogix application.	
 CanBeDeleted	Boolean - Read Only
Returns whether or not the data file may be deleted	
 CanBeMonitored	Boolean - Read Only
Returns whether or not the data file may be monitored.	
 CanChangeScope	Boolean - Read Only
Returns whether or not the scope of this file can be changed.	
 CanChangeSize	Boolean - Read Only
Returns whether or not the file can have the number of elements changed.	
 Debug	Boolean - Read Only
Returns whether or not the file is for debug use only.	
 Description	String - Read/Write
Represents the text description of this data file.	

 FileNumber	Integer - Read Only
Returns the file number of this data file.	
 FormattedName	String - Read Only
Returns the full formatted name of the data file.	
 GlobalScope	Boolean - Read Only
Returns whether or not this data file is of global scope.	
 InUse	Boolean - Read Only
Returns whether or not this file is being used.	
 LocalScope	Boolean - Read Only
Returns whether or not this data file is of local scope.	
 MaxDescriptionLength	Integer - Read Only
Returns the maximum number of characters for the file description.	
 MaxNameLength	Integer - Read Only
Returns the maximum number of characters for the file name.	
 Name	String - Read/Write
Returns or sets the name of the file	
 NumberOfElements	Integer - Read/Write in RSLogix 5 Read Only in RSLogix 500
Returns (or sets, with RSLogix 5 only) the number of elements in this data file	
 Online	Boolean - Read Only
Returns whether or not the data file is online in the processor.	

<i>(5 only)</i>		ReadPrivilege	Boolean - Read Only
This property returns whether or not under the current privilege class the data file is read-enabled. This is a feature available only to processors with the passwords and privileges functionality.			
<i>(500 only)</i>		Reserved	Boolean - Read Only
This property returns True if the data file is reserved.			
		Scopeable	Boolean - Read Only
Returns whether or not this file can be scoped.			
		Type	IgxDataFileTypeConstants - Read Only
Returns the type of data file as a IgxDataFileTypeConstants. The valid selections are listed and defined in Appendix B.			
		TypeAsString	String - Read Only
Returns the type of data file as a text string.			
<i>(5 only)</i>		WritePrivilege	Boolean - Read Only
This property returns whether or not under the current privilege class the data file is write-enabled. This is a feature available only to processors with the passwords and privileges functionality.			

Methods

There are no methods defined for the DataFile object.

Events

There are no events defined for the DataFile object.

Summary Example

Important

This book assumes that you have the basic knowledge required to work with forms and controls in Visual Basic.

The following example automates functionality within RSLogix 5 with the automation interface by incorporating properties, methods and events from the Application and LogixProject object and the DataFiles collection. Comments within the code are preceded by an apostrophe ('). You'll see that although the example is specific to RSLogix 5 software, it is generic enough to adapt to RSLogix 500 with only minor form and comment alterations.

Form

The following form builds on the forms first presented in Chapters 2, 3 and 7.

The screenshot shows a Windows-style application window titled "RSLogix VB Example". The interface is organized into several sections, each with a title bar and a dotted background:

- Application Object:** Contains four buttons: "Start RSLogix5", "Upload", "GetActiveProject", and "Stop RSLogix5".
- LogixProject Object:** Contains two buttons: "Save" and "Download". Below them are two text input fields: "LogixProject.FullName Property" and "LogixProject.Name Property".
- DataFiles Collection:** Contains a "File Number" text input field, two buttons: "Get DataFile" and "Update DataFile fields", and two more text input fields: "Address" and "Value", with a "SetData" button to the right of the "Value" field.
- DataFile Object:** Contains three text input fields: "Number of Elements", "TypeAsString", and "Name".

Code

```
'-----  
' Global variables  
'-----  
  
Dim gApplication As RSLogix5.Application 'Application object  
Dim gProject As RSLogix5.LogixProject 'LogixProject object  
Dim gDataFiles As RSLogix5.DataFiles 'DataFiles Collection  
Dim gDataFile As RSLogix5.DataFile 'DataFile object
```

```

'-----
' Application
'-----

Private Sub Command1_Click()

    ' Set the application object to object returned from CreateObject.
    ' CreateObject is simply a method provided by Microsoft that creates
    ' a new registered COM application instance. In this case we start
    ' RSLogix 5 by using the "RSLogix5.Application" string.
    Set gApplication = CreateObject("RSLogix5.Application")
    ' At this point, if the CreateObject method functioned properly, the
    ' gApplication object is now a direct reference to the RSLogix5
    ' Object Model. Any properties or methods that we invoke on this
    ' object will immediately take effect in RSLogix.

    ' Immediately set the visible property of the application to 'True'
    gApplication.Visible = True

    ' Assign the AutoSaveInterval value to 3 minutes
    gApplication.AutoSaveInterval = 3

    ' Assign WindowState prop to lgxWindowStateMaximized enumeration
    gApplication.WindowState = lgxWindowStateMaximized

End Sub

Private Sub Command2_Click()
    ' Quit the application ignoring prompts and not saving changes.
    gApplication.Quit True, False
    ' Eliminate the reference to the application object
    Set gApplication = Nothing
End Sub

```

```

Private Sub Command3_Click()
On Error GoTo errorHandler
    ' Upload a project from the processor using the upload method of the
    ' application object while ignoring prompts, NOT saving the previous
    ' file, creating new file from the upload (using lgxUploadCreateNew
    ' enum (see objectbrowser for more enumerations)), and going online
    ' (using the lgxGoOnline enum).
    ' Set the returned object reference to the gProject object.
    Set gProject = gApplication.Upload(True, False, lgxUploadCreateNew,
lgxGoOnline)
Exit Sub

errorHandler:
    ' Upon a caught error decide what to do.
    MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Command4_Click()
    ' Get the currently open project in the application.
    Set gProject = gApplication.GetActiveProject
End Sub

'-----
' LogixProject
'-----

Private Sub Command5_Click()
    Dim ReturnValue As Boolean

    ' Save the currently open project, assign return value to a variable
    ' and display that value in a message box.
    ReturnValue = gProject.Save(True, True)
    MsgBox "Returned: " & ReturnValue

End Sub

Private Sub Command6_Click()
On Error GoTo errorHandler
    Dim ReturnValue As Boolean

    ' Download the project to the current processor. This method call is
    ' ignoring all prompts, going online (using the lgxGoOnline enum),
    ' setting the processor to remote program mode (using the
    ' lgxREMOTEPROG enum) and displaying the return value in a message
    ' box
    ReturnValue = gProject.Download(True, lgxGoOnline, lgxREMOTEPROG)
    MsgBox "Returned: " & ReturnValue
Exit Sub

errorHandler:

```

```

        ' Upon a caught error decide what to do.
        MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Text1_Click()
    ' Display the current name of the project in a text box.
    Text1.Text = gProject.FullName
    Text2.Text = gProject.Name
End Sub

'-----
' DataFiles
'-----

Private Sub Command7_Click()
    ' Set the current datafiles reference to a global variable.
    Set gDataFiles = gProject.DataFiles

    ' Retrieve the datafile reference for file number entered
    ' by the user in the text box.
    Set gDataFile = gDataFiles.Item(CInt(Text3.Text))

End Sub

Private Sub Command8_Click()
    Dim ReturnValue As Boolean

    ' Set the current datafiles reference to a global variable.
    Set gDataFiles = gProject.DataFiles

    ' Set a data value at a user specified address to a user specified
    ' value
    ReturnValue = gDataFiles.SetDataValue(Text4.Text, Text5.Text)

End Sub

'-----
' DataFile
'-----

Private Sub Command9_Click()
    ' Set the current datafiles reference to a global variable.
    Set gDataFiles = gProject.DataFiles

    ' Set the DataFile reference specified by the user to the current
    ' global variable.
    Set gDataFile = gDataFiles(CInt(Text3.Text))

    ' Display the values of the NumberOfElements, TypeAsString, and
    ' name properties to their respective text boxes.

```

```
Text6.Text = gDataFile.NumberOfElements
Text7.Text = gDataFile.TypeAsString
Text8.Text = gDataFile.Name
End Sub

Private Sub Text8_Change()
    ' As a user enters a new name into the text field, update the
    ' name property in RSLogix5
    gDataFile.Name = Text8.Text
End Sub
```

Chapter
9

LadderFile object

The LadderFile object represents a ladder file in the project/processor. Obtain the LadderFile object from the “ProgramFiles” Collection. You cannot create a separate instance of the LadderFile object with the CreateObject function.







Properties	Methods	Events
Application	GetRung	-None-
Debug	GetRungAsAscii	
DefaultName	InsertRungAsAscii	
Description	NumerOfRungs	
EditsActive	RemoveRung	
FileNumber		
FormattedName		
InUse		
MaxDescriptionLength		
MaxNameLength		
Name		
Online		
OnlineEdits		
Programmable		
ProtectionSupported (500 only)		
RamEditsPending		
ReadPrivilege (5 only)		
Reserved		
Type		
WritePrivilege (5 only)		


The following commented code gets the ladder file which was specified by the FileNumber variable that was passed in. If this fails an error message is returned.

```
Set gLadderFile = gLogixProject.ProgramFiles(FileNumber)
'get the ladderfile object from RSLogix
If gLadderFile Is Nothing Then
'if that failed then display an error and exit
    MsgBox "ERROR: RSLogix could not get the requested Ladder File",
vbExclamation, "ERROR 008"
    Exit Function
End If
```

Properties

In most cases properties are characteristics or attributes of an object. Using a property returns information about the object or causes a quality of the object to change. The following properties define the LadderFile object.

 Application	Application - Read Only
This property returns the application object.	
 Debug	Boolean - Read Only
This property returns whether or not the ladder file is for debugging use only.	
 DefaultName	String - Read Only
This property returns the default name of the ladder file.	
 Description	String - Read/Write
This property returns or sets the description string for the ladder file.	
 EditsActive	Boolean - Read Only
This property returns whether or not there are any edits active in the processor that have not been assembled	
 FileNumber	Long - Read Only
This property returns the number of the ladder file.	

 **FormattedName** **String - Read Only**

This property returns the formatted name of the file. The format returned is as follows: SYS 0, LAD 2, SFC 4, or STX 6.

 **InUse** **Boolean - Read Only**


This property returns whether or not the file is being used.

 **MaxDescriptionLength** **Long - Read Only**

This property returns the maximum allowable characters for the ladder file description.

 **MaxNameLength** **Long - Read Only**

This property returns the maximum allowable characters for the file name.

 **Name** **String - Read/Write**

This property returns or sets the name for the ladder file.

 **Online** **Boolean - Read Only**

This property returns whether or not the project controlling this ladder file is currently online with the processor.

 **OnlineEdits** **Boolean - Read Only**

This property returns whether or not there are any online edits.

 **Programmable** **Boolean - Read Only**

This property returns whether or not the ladder file is programmable.

(500 only)  **ProtectionSupported** **Boolean - Read Only**

This property returns the attribute of protection supported by this ladder file.

 **RamEditsPending** **Boolean - Read Only**

This property returns whether or not there are any edits pending that are not in the processor or verified in the offline file.

(5 only)



ReadPrivilege**Boolean - Read Only**

This property returns whether or not under the current privilege class the ladder file is read-enabled. This is a feature available only to processors with the passwords and privileges functionality.



Reserved**Boolean - Read Only**

This property returns True if the ladder file is reserved.



Type**IgxProgramFileTypeConstants - Read Only**

This property returns the type of ladder file as a `IgxProgramFileTypeConstants`. The valid selections are listed and defined in Appendix B.

(5 only)



WritePrivilege**Boolean - Read Only**

This property returns whether or not under the current privilege class the ladder (program) file is write-enabled. This is a feature available only to processors with the passwords and privileges functionality.

Methods

Using a method causes something to happen to an object. In most cases methods are actions. Use any of the following methods to identify an action for the `LadderFile` object to perform. Although written for the RSLogix 5 software product, the short examples following each method may be easily adapted to RSLogix 500. For example, type definitions may vary between products, and those differences must be considered when adapting code to the RSLogix 500 object model.



GetRung**Rung**

Use this method to retrieve the specified rung of logic.

Syntax

`GetRung(RungNumber as Long) as Rung`

Arguments

RungNumber - The rung number to get.

Returns

If successful the rung object located at the index `rung` is returned.

Example

The following code snippet gets the current rung of ladder logic (where “`currung`” was initialized as an integer) and sets its value to the Rung object.

```
Set gRung = gLadderFile.GetRung(currung)
```



GetRungAsAscii

String

Use this method to retrieve the ASCII format for a specified rung of ladder logic.

Syntax

`GetRungAsAscii(RungNumber as Long) as String`

Arguments

RungNumber - The rung number to get.

Returns

If successful the ASCII representation of the rung object is returned; otherwise returns a Null string.

Example

The following code snippet gets the ASCII rung text and displays it in a text box.

```
Text1.Text = gLadderFile.RungAsAscii(x)
```



InsertRungAsAscii

Boolean

Use this method to insert a rung or rungs of logic into the ladder file by providing the ASCII format of the rung.

Syntax

`InsertRungAsAscii(RungNumber as Integer, RungString as String) as Boolean`

Arguments

RungNumber as Integer - The rung number to insert.

RungString as String - The ASCII string representing the component makeup of the rung(s) to insert. Make sure to begin each rung in the string with a SOR (start of rung) and end it with an EOR (end of rung) statement.

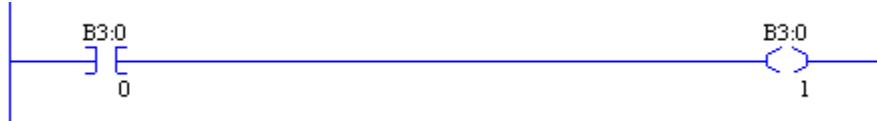
Returns

If successful a rung (or rungs) of ladder logic is inserted into the program file and a value of True is returned; if unsuccessful False is returned.

Example

The following code snippet inserts the following rung at position #4 in your ladder logic program.

```
Res = gLadderFile.InsertRungAsAscii(4, "SOR XIC B3/0 OTE B3/1 EOR")
```



NumberOfRungs

Integer

Use this method to determine the number of rungs in the ladder file.

Syntax

NumberOfRungs() as Integer

Returns

If successful the number of rungs in the file is returned.

Example

The following code snippet sets the variable Y equal to the number of rungs in the ladder logic program.

```
Y = gLadderFile.NumberOfRung
```



RemoveRung

Boolean

Use this method to remove a rung of logic from the ladder file by providing the rung number.

Syntax

RemoveRung(*RungNumber as Long*) as Boolean

Arguments

RungNumber - The number of the rung to be removed.

Returns

If successful a value of True is returned; if unsuccessful False is returned.

Example

The following code snippet removes rung #13 from the ladder logic program.

```
Result = gLadderFile.RemoveRung(13)
```

Events

No events have been defined for the ProgramFiles collection.

Summary example

Important

This book assumes that you have the basic knowledge required to work with forms and controls in Visual Basic.

The following example automates functionality within RSLogix 5 with the automation interface. Comments within the code are preceded by an apostrophe ('). You'll see that although the example is specific to RSLogix 5 software, it is generic enough to adapt to RSLogix 500 with only minor form and comment alterations.

Form

The following form builds on previous examples in this book.

The screenshot shows a Windows-style application window titled "RSLogix VB Example". The window is divided into several sections, each with a label and associated controls:

- Application Object:** Contains three buttons: "Start RSLogix5", "Upload", and "Stop RSLogix5".
- LogixProject Object:** Contains two buttons: "Save" and "Download".
- LogixProject.FullName Property:** A text input field.
- LogixProject.Name Property:** A text input field.
- Processor Object:** Contains a label "Processor.Node" and a text input field.
- Processor Object (continued):** Contains a button "Enable Forces".
- Program Files Collection:** Contains a label "Program File:" followed by a text input field, and three buttons: "Add", "Remove", and "Get". To the right is the label "For ProgramFile:".
- ProgramFile:** Contains a label "Type:" followed by a text input field, a label "Name:" followed by a text input field, and a button "Convert to LadderFile".
- LadderFile:** Contains a label "Rung Number:" followed by a text input field, a label "Rung Ascii:" followed by a text input field, and a button "Add Rung".

Code

```
'-----  
' Global variables  
'-----  
  
Dim gApplication As RSLogix5.Application 'Application object  
Dim gProject As RSLogix5.LogixProject 'LogixProject object  
Dim gProcessor As RSLogix5.Processor 'Processor object  
Dim gProgramFiles As RSLogix5.ProgramFiles 'ProgramFiles object  
Dim gProgramFile As RSLogix5.ProgramFile 'ProgramFile object  
Dim gLadderFile As RSLogix5.LadderFile 'LadderFile Object  
  
'-----  
' Application  
'-----  
  
Private Sub Command1_Click()  
  
    ' Set application object to the object returned from CreateObject.  
    ' CreateObject is simply a method provided by Microsoft that creates  
    ' a new registered COM application instance. In this case we start  
    ' RSLogix 5 by using the "RSLogix5.Application" string.  
    Set gApplication = CreateObject("RSLogix5.Application")  
    ' At this point, if the CreateObject method functioned properly, the  
    ' gApplication object is now a direct reference to the RSLogix5  
    ' Object Model. Any properties or methods that we invoke on this  
    ' object will immediately take effect in RSLogix.  
  
    ' Immediately set the visible property of the application to 'True'  
    gApplication.Visible = True  
  
    ' Assign the AutoSaveInterval value to 3 minutes  
    gApplication.AutoSaveInterval = 3  
  
    ' Assign WindowState prop to lgxWindowStateMaximized enumeration  
    gApplication.WindowState = lgxWindowStateMaximized  
  
End Sub  
  
Private Sub Command2_Click()  
    ' Quit the application ignoring prompts and not saving changes.  
    gApplication.Quit True, False  
    ' Eliminate the reference to the application object  
    Set gApplication = Nothing  
End Sub  
  
Private Sub Command3_Click()  
On Error GoTo errorHandler  
    ' Upload a project from the processor using the upload method of the  
    ' application object while ignoring prompts, NOT saving the previous
```

```

' file, creating new file from the upload (using lgxUploadCreateNew
' enum (see objectbrowser for more enumerations)), and going online
' (using the lgxGoOnline enum).
' Set the returned object reference to the gProject object.
Set gProject = gApplication.Upload(True, False, lgxUploadCreateNew,
lgxGoOnline)
Exit Sub

errorHandler:
' Upon a caught error decide what to do.
MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Command4_Click()
' Get the currently open project in the application.
Set gProject = gApplication.GetActiveProject
End Sub

'-----
' LogixProject
'-----

Private Sub Command5_Click()
Dim ReturnValue As Boolean

' Save the currently open project, assign the return value to a
' variable and display that value in a message box.
ReturnValue = gProject.Save(True, True)
MsgBox "Returned: " & ReturnValue

End Sub

Private Sub Command6_Click()
On Error GoTo errorHandler
Dim ReturnValue As Boolean

' Download the project to the current processor. This method call is
' ignoring all prompts, going online (using the lgxGoOnline enum),
' setting the processor to remote program mode (using the
' lgxREMOTEPROG enum) and displaying the return value in a message
' box
ReturnValue = gProject.Download(True, lgxGoOnline, lgxREMOTEPROG)
MsgBox "Returned: " & ReturnValue
Exit Sub

errorHandler:
' Upon a caught error decide what to do.
MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

```



```

Private Sub Text1_Click()
    ' Display the current name of the project in a text box.
    Text1.Text = gProject.FullName
    Text2.Text = gProject.Name
End Sub

'-----
' Processor
'-----

Private Sub Text3_Click()
    ' Set the processor's reference to a global variable.
    Set gProcessor = gProject.Processor

    ' Display the current node in a text box.
    Text3.Text = gProcessor.Node
End Sub

Private Sub Command7_Click()
    ' Set the processor's reference to a global variable.
    Set gProcessor = gProject.Processor

    ' Enable forces in the processor
    gProcessor.EnableForces
End Sub

'-----
' Program Files Collection
'-----

Private Sub Command8_Click()
    ' Set the current programfiles collection reference from the project
    ' to a global variable.
    Set gProgramFiles = gProject.ProgramFiles

    ' Add new ladder file into the ProgramFiles collection. This method
    ' call sets the gProgramFile object to a new ProgramFile object
    ' created at the file number specified by the value of text4.text,
    ' using the lgxLADDER enum to specify to create a ladder file,
    ' that is NOT a debug file, and ignoring all prompts.
    Set gProgramFile = gProgramFiles.Add(CInt(Text4.Text), lgxLADDER,
    False, True)

End Sub

Private Sub Command9_Click()
    Dim ReturnValue As Boolean

    ' Set the current programfiles collection reference from the project

```

```

' to a global variable.
Set gProgramFiles = gProject.ProgramFiles

' Remove the ProgramFile specified in the ext box from the
' ProgramFiles collection.
' Display the returned value in a message box.
ReturnValue = gProgramFiles.Remove(CInt(Text4.Text), True)
MsgBox "Returned: " & ReturnValue

End Sub

'-----
' ProgramFile Object
'-----

Private Sub Command10_Click()

' Set the current programfiles collection reference from the project
' to a global variable.
Set gProgramFiles = gProject.ProgramFiles

' Set the programfile reference specified by the value of a textbox
' to the current global object.
Set gProgramFile = gProgramFiles(CInt(Text4.Text))

End Sub

Private Sub Text5_Click()
' Assign the textboxes the Type and Name of the programfile.
Text5.Text = gProgramFile.Type
Text6.Text = gProgramFile.Name
End Sub

Private Sub Text6_Change()
' Change the ProgramFile.Name property to the current text.
gProgramFile.Name = Text6.Text
End Sub

'-----
' LadderFile Object
'-----

Private Sub Command11_Click()
' Cast the current program object to a LadderFile object.
Set gLadderFile = gProgramFile

End Sub

Private Sub Command12_Click()
Dim ReturnValue As Boolean

```

```
' Add a rung defined by user entered text and display the return  
' value in a message box.  
ReturnValue = gLadderFile.InsertRungAsAscii(CInt(Text8.Text) ,  
Text7.Text)  
MsgBox "Returned: " & ReturnValue  
End Sub
```


Chapter 10 Rung object

The Rung object represents a rung of ladder logic. Obtain the Rung object from the LadderFile object using the GetRung method. You cannot create a separate instance of the Rung object with the CreateObject function.










Properties	Methods	Events
Active	-None-	-None-
Application		
Comment		
DbaseID		
EditsActive		
EndRung		
FileNumber		
Modified		
NumberOfInstructions		
Online		
Output		
RungNumber		
RungType		
RungZoneDisplay		
TempReplace		
Title		
Verified		

The following code example illustrates how you might access the Rung object.

```
gRung = gLadderFile.GetRung(Rung_Number)
If gRung Is Nothing Then
    MsgBox "Rung not valid"
End If
```

Properties

In most cases properties are characteristics or attributes of an object. Using a property returns information about the object or causes a quality of the object to change. The following properties define the Rung object.

 Active	Boolean - Read Only
This property returns whether or not this rung is active.	
 Application	Application - Read Only
This property returns the Application object.	
 Comment	String - Read Only
This property returns the rung comment associated with the rung.	
 DbaseID	Long - Read Only
This property returns the ID used to retrieve the rung title and comment from the database.	
 EditsActive	Boolean - Read Only
This property returns whether or not this rung contains edits that are active in the processor.	
 EndRung	Boolean - Read Only
This property returns whether or not this is the end rung in the ladder file.	
 FileNumber	Long - Read Only
This property returns the number of the file in which this rung resides.	
 Modified	Boolean - Read Only
This property returns whether or not this rung has been modified in any way.	
 NumberOfInstructions	Long - Read Only
This property returns how many instructions there are on the rung.	



Online**Boolean - Read Only**

This property returns whether or not the project containing this rung is currently online with the processor.



Output**Boolean - Read Only**

This property returns whether or not there is an output instruction on this rung.



RungNumber**Long - Read Only**

This property returns the number of the rung.



RungType**IgxRungZoneTypes - Read Only**

This property returns what type of rung the specified rung is. Possible returned types are listed below and described in Appendix B.

- (0) IgxPlainRung
- (1) IgxReplaceRung
- (2) IgxInsertRung
- (3) IgxDeleteRung
- (4) IgxEditRung



RungZoneDisplay**IgxRungZoneTypes - Read Only**

This property returns the type of rung adjustment (if any) that is currently applied to the rung. Possible returned types are listed below and described in Appendix B.

- (0) IgxPlainRung
- (1) IgxReplaceRung
- (2) IgxInsertRung
- (3) IgxDeleteRung
- (4) IgxEditRung
- (5) IgxTmpInsertRung
- (6) IgxTmpReplaceRung
- (7) IgxAnyIrdRung



TempReplace**Boolean - Read Only**

This property returns whether or not this is a temporary replacement rung (marked with an R zone marker).



Title**String - Read Only**

This property returns the rung title associated with the rung.



Verified**Boolean - Read Only**

This property returns whether or not the rung has been verified.

Methods

There are no Methods defined for the Rung object.

Events

There are no Events defined for the Rung object.

Summary example

Important

This book assumes that you have the basic knowledge required to work with forms and controls in Visual Basic.

The following example automates functionality within RSLogix 5 with the automation interface. Comments within the code are preceded by an apostrophe ('). You'll see that although the example is specific to RSLogix 5 software, it is generic enough to adapt to RSLogix 500 with only minor form and comment alterations.

Form

The following form builds on previous examples in this book.

The screenshot shows a Windows-style application window titled "RSLogix VB Example". The form is organized into several sections, each with a dotted background:

- Application Object:** Contains buttons for "Start RSLogix5", "Upload", "GetActiveProject", and "Stop RSLogix5".
- LogixProject Object:** Contains buttons for "Save" and "Download", and two text input fields labeled "LogixProject.FullName Property:" and "LogixProject.Name Property:".
- Processor Object:** Contains a text input field for "Processor.Node" and an "Enable Forces" button.
- Program Files Collection:** Contains a "Program File" text input, "Add", "Remove", and "Get" buttons, and the text "For ProgramFile:".
- ProgramFile:** Contains "Type:" and "Name" text inputs, and a "Convert to LadderFile" button.
- LadderFile:** Contains "Rung Number" and "Rung Ascii" text inputs, and an "Add Rung" button.
- Rung:** Contains "Rung Number" text input and a "Get" button, followed by "Rung Title" and "Rung Comment" text input fields.

Code

```
'-----  
' Global variables  
'-----  
  
Dim gApplication As RSLogix5.Application 'Application object  
Dim gProject As RSLogix5.LogixProject 'LogixProject object  
Dim gProcessor As RSLogix5.Processor 'Processor object  
Dim gProgramFiles As RSLogix5.ProgramFiles 'ProgramFiles object  
Dim gProgramFile As RSLogix5.ProgramFile 'ProgramFile object  
Dim gLadderFile As RSLogix5.LadderFile 'LadderFile Object  
Dim gRung As RSLogix5.Rung 'Rung Object  
  
'-----  
' Application  
'-----  
  
Private Sub Command1_Click()  
  
    ' Set application object to the object returned from CreateObject.  
    ' CreateObject is simply a method provided by Microsoft that creates  
    ' a new registered COM application instance. In this case we start  
    ' RSLogix 5 by using the "RSLogix5.Application" string.  
    Set gApplication = CreateObject("RSLogix5.Application")  
    ' At this point, if the CreateObject method functioned properly, the  
    ' gApplication object is now a direct reference to the RSLogix5  
    ' Object Model. Any properties or methods that we invoke on this  
    ' object will immediately take effect in RSLogix.  
  
    ' Immediately set the visible property of the application to 'True'  
    gApplication.Visible = True  
  
    ' Assign the AutoSaveInterval value to 3 minutes  
    gApplication.AutoSaveInterval = 3  
  
    ' Assign WindowState prop to lgxWindowStateMaximized enumeration  
    gApplication.WindowState = lgxWindowStateMaximized  
  
End Sub  
  
Private Sub Command2_Click()  
    ' Quit the application ignoring prompts and not saving changes.  
    gApplication.Quit True, False  
    ' Eliminate the reference to the application object  
    Set gApplication = Nothing  
End Sub  
  
Private Sub Command3_Click()  
On Error GoTo errorHandler  
    ' Upload a project from the processor using the upload method of the
```

```

    ' application object while ignoring prompts, NOT saving the previous
    ' file, creating new file from the upload (using lgxUploadCreateNew
    ' enum (see objectbrowser for more enumerations)), and going online
    ' (using the lgxGoOnline enum).
    ' Set the returned object reference to the gProject object.
    Set gProject = gApplication.Upload(True, False, lgxUploadCreateNew,
lgxGoOnline)
Exit Sub

errorHandler:
    ' Upon a caught error decide what to do.
    MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Command4_Click()
    ' Get the currently open project in the application.
    Set gProject = gApplication.GetActiveProject
End Sub

'-----
' LogixProject
'-----

Private Sub Command5_Click()
    Dim ReturnValue As Boolean

    ' Save the currently open project, assign the return value to a
    ' variable and display that value in a message box.
    ReturnValue = gProject.Save(True, True)
    MsgBox "Returned: " & ReturnValue

End Sub

Private Sub Command6_Click()
On Error GoTo errorHandler
    Dim ReturnValue As Boolean

    ' Download the project to the current processor. This method call is
    ' ignoring all prompts, going online (using the lgxGoOnline enum),
    ' setting the processor to remote program mode (using the
    ' lgxREMOTEPROG enum) and displaying the return value in a message
    ' box
    ReturnValue = gProject.Download(True, lgxGoOnline, lgxREMOTEPROG)
    MsgBox "Returned: " & ReturnValue
Exit Sub

errorHandler:
    ' Upon a caught error decide what to do.
    MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description

```

```

End Sub

Private Sub Text1_Click()
    ' Display the current name of the project in a text box.
    Text1.Text = gProject.FullName
    Text2.Text = gProject.Name
End Sub

'-----
' Processor
'-----

Private Sub Text3_Click()
    ' Set the processor's reference to a global variable.
    Set gProcessor = gProject.Processor

    ' Display the current node in a text box.
    Text3.Text = gProcessor.Node
End Sub

Private Sub Command7_Click()
    ' Set the processor's reference to a global variable.
    Set gProcessor = gProject.Processor

    ' Enable forces in the processor
    gProcessor.EnableForces
End Sub

'-----
' Program Files Collection
'-----

Private Sub Command8_Click()
    ' Set the current programfiles collection reference from the project
    ' to a global variable.
    Set gProgramFiles = gProject.ProgramFiles

    ' Add new ladder file into the ProgramFiles collection. This method
    ' call sets the gProgramFile object to a new ProgramFile object
    ' created at the file number specified by the value of text4.text,
    ' using the lgxLADDER enum to specify to create a ladder file,
    ' that is NOT a debug file, and ignoring all prompts.
    Set gProgramFile = gProgramFiles.Add(CInt(Text4.Text), lgxLADDER,
    False, True)

End Sub

Private Sub Command9_Click()
    Dim ReturnValue As Boolean

    ' Set the current programfiles collection reference from the project

```

```

' to a global variable.
Set gProgramFiles = gProject.ProgramFiles

' Remove the ProgramFile specified in the ext box from the
' ProgramFiles collection.
' Display the returned value in a message box.
ReturnValue = gProgramFiles.Remove(CInt(Text4.Text), True)
MsgBox "Returned: " & ReturnValue

End Sub

'-----
' ProgramFile Object
'-----

Private Sub Command10_Click()

' Set the current programfiles collection reference from the project
' to a global variable.
Set gProgramFiles = gProject.ProgramFiles

' Set the programfile reference specified by the value of a textbox
' to the current global object.
Set gProgramFile = gProgramFiles(CInt(Text4.Text))

End Sub

Private Sub Text5_Click()
' Assign the textboxes the Type and Name of the programfile.
Text5.Text = gProgramFile.Type
Text6.Text = gProgramFile.Name

End Sub

Private Sub Text6_Change()
' Change the ProgramFile.Name property to the current text.
gProgramFile.Name = Text6.Text

End Sub

'-----
' LadderFile Object
'-----

Private Sub Command11_Click()
' Cast the current program object to a LadderFile object.
Set gLadderFile = gProgramFile

End Sub

Private Sub Command12_Click()

```

```

Dim ReturnValue As Boolean

' Add a rung defined by user entered text and display the return
' value in a message box.
ReturnValue = gLadderFile.InsertRungAsAscii(CInt(Text8.Text),
Text7.Text)
MsgBox "Returned: " & ReturnValue
End Sub

'-----
' Rung Object
'-----

Private Sub Command13_Click()
' Set the global rung object to the rung number specified by a
' textbox.
Set gRung = gLadderFile.GetRung(CInt(Text11.Text))

' Assign the rung title and comment property to their respective
' textboxes.
Text9.Text = gRung.Title
Text10.Text = gRung.Comment

End Sub

```

RevisionNotes object

The RevisionNotes object represents the revision notes associated with any project. Obtain the RevisionNotes object from the LogixProject object via the RevisionNotes property. The RevisionNotes object can not be created with the CreateObject function.




Properties	Methods	Events
Application	Count	-None-
InternalRevision	RevisionNote	
Revision		

The following commented code example illustrates how you might get all the revision note information associated with a project. The example further adds error checking and displays a message if nothing is returned.

```
'This function calls RSLogix to get all Revision Note information
'from RSLogix
Private Function GetRevisionNotes()
'get the RevisionNotes object from the LogixProject Object
Set gRevisionNotes = gLogixProject.RevisionNotes
If gRevisionNotes Is Nothing Then
'if RSLogix fails to get the object then exit
MsgBox "ERROR: Could not get Revision information",
vbExclamation, "ERROR 004"
Exit Function
End If
End Function
```


Properties

In most cases properties are characteristics or attributes of an object. Using a property returns information about the object or causes a quality of the object to change. The following properties define the RevisionNotes object.

 Application	Application - Read Only
This property returns the Application object.	
 InternalRevision	Integer - Read Only
This property returns the internal revision number of the parent project. This revision number is incremental and does not roll over at 999, but rather reflects the latest sequential revision of the project.	
 Revision	Long - Read Only
This property returns the revision of the project as reflected on the Revision History/Editor dialog. This is a number between 0 and 999.	

Methods

Using a method causes something to happen to an object. In most cases methods are actions. Use any of the following methods to identify an action for the RevisionNotes object to perform. Although written for the RSLogix 5 software product, the short examples following each method may be easily adapted to RSLogix 500. For example, type definitions may vary between products, and those differences must be considered when adapting code to the RSLogix 500 object model.

 Count	Integer
Use this method to read how many revision notes have been saved with the project.	
Syntax	
Count() As Integer	
Returns	
If successful the number of revision notes is returned.	
Example	
The following code snippet displays the number of revision notes associated with your project.	


```
MsgBox "There are " & gRevisionNotes.Count & " revision notes."
```



RevisionNote

String

Use this method to return the text of a revision note associated with a project by providing the number of the revision note.

Syntax

RevisionNote(*NoteNumber As Long*) As String

Arguments

NoteNumber - The number of a specific revision note in the project.

Returns

If successful the specified revision note text string is returned.

Example

The following code snippet returns the text of the first revision note saved with the project.

```
Dim value as String  
value = gRevisionNotes.RevisionNote(0)
```

Events

There are no events defined for the RevisionNotes object.

Summary example

Important

This book assumes that you have the basic knowledge required to work with forms and controls in Visual Basic.

The following example automates functionality within RSLogix 5 with the automation interface. Comments within the code are preceded by an apostrophe ('). You'll see that although the example is specific to RSLogix 5 software, it is generic enough to adapt to RSLogix 500 with only minor form and comment alterations.

Form

The following form builds on previous examples in this book.

The screenshot shows a Windows application window titled "RSLogix VB Example". The window is divided into three main sections:

- Application Object:** Contains four buttons: "Start RSLogix5", "Upload", "GetActiveProject", and "Stop RSLogix5".
- LogixProject Object:** Contains two buttons: "Save" and "Download". Below these are two text input fields: "LogixProject.FullName Property" and "LogixProject.Name Property".
- Revision Notes:** Contains a table with two columns: "Number" and "Revision Note". To the right of the table is a "Get" button. Below the table is a large text area for entering notes.

Code

```
'-----  
' Global variables  
'-----  
  
Dim gApplication As RSLogix5.Application 'Application object  
Dim gProject As RSLogix5.LogixProject 'LogixProject object  
Dim gRevisionNotes As RSLogix5.RevisionNotes 'RevisionNotes Object  
  
'-----  
' Application  
'-----  
  
Private Sub Command1_Click()  
  
    ' Set application object to the object returned from CreateObject.  
    ' CreateObject is simply a method provided by Microsoft that creates  
    ' a new registered COM application instance. In this case we start  
    ' RSLogix 5 by using the "RSLogix5.Application" string.
```

```

Set gApplication = CreateObject("RSLogix5.Application")
' At this point, if the CreateObject method functioned properly, the
' gApplication object is now a direct reference to the RSLogix5
' Object Model. Any properties or methods that we invoke on this
' object will immediately take effect in RSLogix.

' Immediately set the visible property of the application to 'True'
gApplication.Visible = True

' Assign the AutoSaveInterval value to 3 minutes
gApplication.AutoSaveInterval = 3

' Assign WindowState prop to lgxWindowStateMaximized enumeration
gApplication.WindowState = lgxWindowStateMaximized
End Sub

Private Sub Command2_Click()
' Quit the application ignoring prompts and not saving changes.
gApplication.Quit True, False
' Eliminate the reference to the application object
Set gApplication = Nothing
End Sub

Private Sub Command3_Click()
On Error GoTo errorHandler
' Upload a project from the processor using the upload method of the
' application object while ignoring prompts, NOT saving the previous
' file, creating a new file from the upload (using the
' lgxUploadCreateNew enum (see the objectbrowser for more
' enumerations)), and go online (using the lgxGoOnline enum).
' Set the returned object reference to the gProject object.
Set gProject = gApplication.Upload(True, False, lgxUploadCreateNew,
lgxGoOnline)
Exit Sub

errorHandler:
' Upon a caught error decide what to do.
MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Command4_Click()
' Get the currently open project in the application.
Set gProject = gApplication.GetActiveProject
End Sub

```

```

'-----
' LogixProject
'-----

Private Sub Command5_Click()
    Dim returnValue As Boolean

    ' Save the currently open project, assign the return value to a
    ' variable and display that value in a message box.
    returnValue = gProject.Save(True, True)
    MsgBox "Returned: " & returnValue
End Sub

Private Sub Command6_Click()
On Error GoTo errorHandler
    Dim returnValue As Boolean

    ' Download the project to the current processor. This method call is
    ' ignoring all prompts, going online (using the lgxGoOnline enum),
    ' setting the processor to remote program mode (using the
    ' lgxREMOTEPROG enum) and displaying the return value in a message
    ' box
    returnValue = gProject.Download(True, lgxGoOnline, lgxREMOTEPROG)
    MsgBox "Returned: " & returnValue
Exit Sub

errorHandler:
    ' Upon a caught error decide what to do.
    MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Text1_Click()
    ' Display the current name of the project in a text box.
    Text1.Text = gProject.FullName
    Text2.Text = gProject.Name
End Sub

'-----
' RevisionNotes
'-----

Private Sub Command7_Click()
On Error GoTo errhand
    ' Set the current revision notes object to a global variable.
    Set gRevisionNotes = gProject.RevisionNotes

    ' Query a user specified revision note and place the returned text
    ' within another textbox.
    Text4.Text = gRevisionNotes.RevisionNote(CInt(Text3.Text))
Exit Sub

```

```
errhand:  
    MsgBox "That revision note was not found."  
End Sub
```


ReportOptions object

The ReportOptions object represents the report settings associated with the project. Obtain the ReportOptions object from the LogixProject object via the ReportOptions property. You cannot create an instance of the ReportOptions object with the CreateObject function.

Properties	Methods	Events
AddressSymbols	-None-	-None-
Application		
ChannelConfiguration		
CrossReference		
CrossReferenceByAddress		
CrossReferenceFileEnd		
CrossReferenceFileStart		
CrossReferenceSymbolEnd		
CrossReferenceSymbolStart		
CustomDataMonitorFileRange		
CustomDataMonitorFiles		
DataFileList		
DataFileRange		
DataFiles		
InstructionComments		
IOInfo		
MemoryUsage		
MemoryUsageFileRange		
Multipoint (<i>RSLogix 500 only</i>)		
ProcessorInfo		
ProgramFileList		
ProgramFileRange		
ProgramFiles		
SymbolGroups		
TitlePage		

The following commented code example illustrates how to access the Report Options object.

```
Set gReportOptions = gLogixProject.ReportOptions
'get a copy of the Report options object
If gReportOptions Is Nothing Then
'if the copy failed then display an error and exit
    MsgBox "Error Getting Report options!", vbExclamation, "ERROR"
    Exit Function
End Sub
```

Properties

In most cases properties are characteristics or attributes of an object. Using a property returns information about the object or causes a quality of the object to change. The following properties define the ReportOptions object.



AddressSymbols

Boolean - Read/Write

If set to True the Address/Symbol Report is selected for inclusion in your printed project report. The Address/Symbol Report provides address, symbol, scope and description information for the addresses in the report.



Application

Application - Read Only

This property returns the Application object.



ChannelConfiguration

Boolean - Read/Write

If set to True the Channel Configuration Report is selected for inclusion in your printed project report. The Channel Configuration Report contains information about how the processor's channels have been configured to communicate.



CrossReference

Boolean - Read/Write

If set to True the Cross Reference Report is selected for inclusion in your printed project report. The Cross Reference Report tells you in which files and rungs the addresses in your project are being used.

 **CrossReferenceByAddress** Boolean - Read/Write

If set to True the Cross Reference Report that you have selected to print will be sorted by address. The sorted listing is of the addresses used in your project, how they are being used, and the location of each address. The list is sorted by data tables. If set to False the Cross Reference Report will be sorted by symbol.

 **CrossReferenceFileEnd** String - Read/Write

This property sets or returns the ending address for inclusion in the cross reference report. This corresponds to the Cross Reference Range which defaults to include all possible data table addresses in the project (data table addresses from 0-255). If you want only certain data table addresses included in the cross reference report, indicate the ending address using this property.

 **CrossReferenceFileStart** String - Read/Write

This property sets or returns the starting address for inclusion in the cross reference report. This corresponds to the Cross Reference Range which defaults to include all possible data table addresses in the project (data table addresses from 0-255). If you want only certain data table addresses included in the cross reference report, indicate the starting address using this property.

 **CrossReferenceSymbolEnd** String - Read/Write

This property sets or returns the ending symbol for inclusion in the cross reference report. This corresponds to the Cross Reference Range which defaults to include all possible data table addresses in the project (data table symbols from A-ZZ). If you want only certain data table addresses included in the cross reference report, indicate the ending symbol using this property.

 **CrossReferenceSymbolStart** String - Read/Write

This property sets or returns the starting symbol for inclusion in the cross reference report. This corresponds to the Cross Reference Range which defaults to include all possible data table addresses in the project (data table symbols from A-ZZ). If you want only certain data table addresses included in the cross reference report, indicate the starting symbol using this property.

 **CustomDataMonitorFileRange** **String - Read/Write**

This range specifies which CDM files are to be printed. The format can be any of the following:

- File numbers separated by a comma: “2,5,10”
- File number range separated by a dash: “2-12”
- A combination of both: “2,3,5-12,25”
- Text specifying all files: “ALL”

 **CustomDataMonitorFiles** **Boolean - Read/Write**

If set to True the Custom Data Monitor Report is selected for inclusion in your printed project report. A CDM report provides you with a list of the addresses in the Custom Data Monitor, their symbols, and the current value of the bit or word address. By default all CDM files are selected for your report. Use CustomDataMonitorFileRange to selectively define the CDM files for inclusion.

 **DataFileList** **Boolean - Read/Write**

If set to True a list of the data files in the project will be included in your printed report.

 **DataFileRange** **String - Read/Write**

This range specifies which data files are selected for inclusion in your printed project report. The format can be any of the following:

- File numbers separated by a comma: “2,5,10”
- File number range separated by a dash: “2-12”
- A combination of both: “2,3,5-12,25”
- Text specifying all files: “ALL”

 **DataFiles** **Boolean - Read/Write**

If set to True the Data Files Report is included in your project reports. By default all the data files in your project are selected for your report. Use DataFileRange to selectively define the data files for inclusion.

 **InstructionComments** **Boolean - Read/Write**

If set to True the Instruction Comments Report is selected for inclusion in your printed project report. This report contains all the instruction comments defined in the database.



IOInfo**Boolean - Read/Write**

If set to True the IO Configuration Report is selected for inclusion in your printed project reports. The IO Configuration Report contains information about the IO modules assembled in your system configuration. Slot #, Part #, description information and the number of input and output words used in each module are included.



MemoryUsage**Boolean - Read/Write**

If set to True the Memory Usage Report is selected for inclusion in your printed project report. The Memory Usage Report provides information about which addresses are used in your project and how they are used. By default all your data files will be included in the Memory Usage report. To individually select data files for inclusion in this report also use the MemoryUsageFileRange property.



MemoryUsageFileRange **String - Read/Write**

This specifies which data file addresses are to be included in the Memory Usage report in your printed project report by providing a numeric range. The format can be any of the following:

- File numbers separated by a comma: “2,5,10”
- File number range separated by a dash: “2-12”
- A combination of both: “2,3,5-12,25”
- Text specifying all files: “ALL”

(500 only)

Multipoint**Boolean - Read/Write**

If set to True the Multipoint Monitor will be included in the report.



ProcessorInfo**Boolean - Read/Write**

If set to True the Processor Information Report is selected for inclusion in your printed project reports. The Processor report includes type, memory used and file content



ProgramFileList**Boolean - Read/Write**

If set to True a Program File List Report is selected for inclusion in your printed project reports.

 **ProgramFileRange** **String - Read/Write**

This range specifies which program files are selected for inclusion in your printed project reports. The format may be any of the following:

- File numbers separated by a comma: “2,5,10”
- File number range separated by a dash: “2-12”
- A combination of both: “2,3,5-12,25”
- Text specifying all files: “ALL”

 **ProgramFiles** **Boolean - Read/Write**

If set to True the Program Files Report is included in your project reports. Use ProgramFileRange to further define the program files for inclusion.

 **SymbolGroups** **Boolean - Read/Write**

If set to True the Symbol Groups Report is selected for inclusion in your printed project report. This report contains the group name and description for all the symbol groups defined in the documentation database.

 **TitlePage** **Boolean - Read/Write**

If set to True a Title Page will be included in your printed project report. This title page is not customizable via the automation interface. It will by default include your project name and the Rockwell Software RSLogix application logo.

Methods

There are no methods defined for the ReportOptions object.

Events

There are no events defined for the ReportOptions object.

Summary example

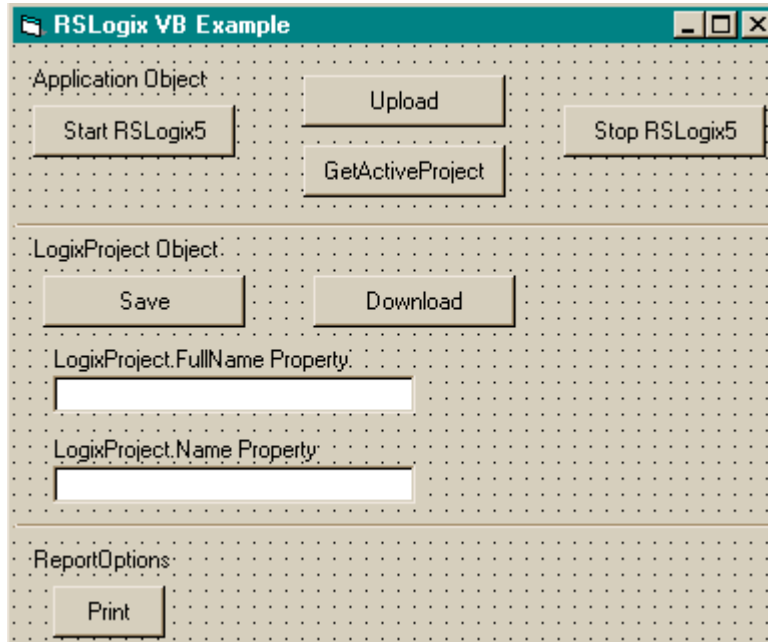
Important

This book assumes that you have the basic knowledge required to work with forms and controls in Visual Basic.

The following example automates functionality within RSLogix 5 with the automation interface. Comments within the code are preceded by an apostrophe ('). You'll see that although the example is specific to RSLogix 5 software, it is generic enough to adapt to RSLogix 500 with only minor form and comment alterations.

Form

The following form builds on previous examples in this book.



Code

```
'-----  
' Global variables  
'-----  
  
Dim gApplication As RSLogix5.Application 'Application object  
Dim gProject As RSLogix5.LogixProject 'LogixProject object  
Dim gReportOptions As RSLogix5.ReportOptions 'ReportOptions Object
```

```

'-----
' Application
'-----

Private Sub Command1_Click()

    ' Set the application object to object returned from CreateObject.
    ' CreateObject is simply a method provided by Microsoft that creates
    ' a new registered COM application instance. In this case we start
    ' RSLogix 5 by using the "RSLogix5.Application" string.
    Set gApplication = CreateObject("RSLogix5.Application")
    ' At this point, if the CreateObject method functioned properly, the
    ' gApplication object is now a direct reference to the RSLogix5
    ' Object Model. Any properties or methods that we invoke on this
    ' object will immediately take effect in RSLogix.

    ' Immediately set the visible property of the application to 'True'
    gApplication.Visible = True

    ' Assign the AutoSaveInterval value to 3 minutes
    gApplication.AutoSaveInterval = 3

    ' Assign WindowState prop to lgxWindowStateMaximized enumeration
    gApplication.WindowState = lgxWindowStateMaximized

End Sub

Private Sub Command2_Click()

    ' Quit the application ignoring prompts and not saving changes.
    gApplication.Quit True, False
    ' Eliminate the reference to the application object
    Set gApplication = Nothing

End Sub

Private Sub Command3_Click()
On Error GoTo errorHandler

    ' Upload a project from the processor using the upload method of the
    ' application object while ignoring prompts, NOT saving the previous
    ' file, create a new file from the upload (using lgxUploadCreateNew
    ' enum (see the objectbrowser for more enumerations)), and going
    ' online (using the lgxGoOnline enum).
    ' Set the returned object reference to the gProject object.
    Set gProject = gApplication.Upload(True, False, lgxUploadCreateNew,
lgxGoOnline)
Exit Sub

```

```

errorHandler:
    ' Upon a caught error decide what to do.
    MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Command4_Click()
    ' Get the currently open project in the application.
    Set gProject = gApplication.GetActiveProject
End Sub

'-----
' LogixProject
'-----

Private Sub Command5_Click()
    Dim ReturnValue As Boolean

    ' Save the currently open project, assign the return value to a
    ' variable and display that value in a message box.
    ReturnValue = gProject.Save(True, True)
    MsgBox "Returned: " & ReturnValue

End Sub

Private Sub Command6_Click()
On Error GoTo errorHandler
    Dim ReturnValue As Boolean

    ' Download the project to the current processor. This method call is
    ' ignoring all prompts, going online (using the lgxGoOnline enum),
    ' setting the processor to remote program mode (using the
    ' lgxREMOTEPROG enum) and displaying the return value in a message
    ' box
    ReturnValue = gProject.Download(True, lgxGoOnline, lgxREMOTEPROG)
    MsgBox "Returned: " & ReturnValue
Exit Sub

errorHandler:
    ' Upon a caught error decide what to do.
    MsgBox "Error: " & Err.Number & vbCrLf & "Description: " &
Err.Description
End Sub

Private Sub Text1_Click()
    ' Display the current name of the project in a text box.
    Text1.Text = gProject.FullName

```

```

        Text2.Text = gProject.Name
    End Sub

'-----
' ReportOptions
'-----

Private Sub Command7_Click()
    Dim ReturnValue As Boolean

    ' Set the current ReportOptions object to a global variable.
    Set gReportOptions = gProject.ReportOptions

    ' Print the current project with all reporting options enabled.
    gReportOptions.AddressSymbols = True
    gReportOptions.ChannelConfiguration = True
    gReportOptions.CrossReference = True
    gReportOptions.CustomDataMonitorFiles = True
    gReportOptions.DataFileList = True
    gReportOptions.DataFiles = True
    gReportOptions.InstructionComments = True
    gReportOptions.IOInfo = True
    gReportOptions.MemoryUsage = True
    gReportOptions.ProcessorInfo = True
    gReportOptions.ProgramFileList = True
    gReportOptions.ProgramFiles = True
    gReportOptions.SymbolGroups = True
    gReportOptions.TitlePage = True
    ' Print the report via the project object and display the return
    ' value in a messagebox.
    ReturnValue = gProject.PrintReport(True)
    MsgBox "Returned: " & ReturnValue

End Sub

```


13 AddrSymRecords collection

The AddrSymRecords collection represents the collection of Address/Symbol database records (AddrSymRecord) in the RSLogix project. The AddrSymRecords collection can be obtained using the AddrSymRecords property of the LogixProject object. The AddrSymRecords collection is not creatable with the CreateObject function.



Properties	Methods	Events
Application	Add	-None-
Count	Duplicate	
	GetRecordIndexViaAddrOrSym	
	GetRecordViaAddrOrSym	
	GetRecordViaDesc	
	GetRecordViaIndex	
	RemoveRecordViaAddrOrSym	
	RemoveRecordViaIndex	
	SearchAndReplaceDesc	

The following commented code example illustrates how you might get the AddrSymRecords collection from the LogixProject object. This example adds error checking and notification.

```
'get the AddrSymRecords collection from the LogixProject object
Set gAddrSymRecords = gLogixProject.AddrSymRecords
'if Logix failed to get the Address/Symbols collection then display
'error and exit
If gAddrSymRecords Is Nothing Then
    MsgBox "ERROR: Could not get Address Symbol Records!",
        vbExclamation, "ERROR"
    Exit Function
End If
```


Properties

In most cases properties are characteristics or attributes of an object. Using a property returns information about the object or causes a quality of the object to change. The following properties define the AddrSymRecords collection.

 Application	Application - Read Only
<hr/>	
This property returns an Application that represents the RSLogix application.	
 Count	Long - Read Only
<hr/>	
This property returns the number of records in the Address/Symbol database.	

Methods

Using a method causes something to happen to an object. In most cases methods are actions. Use any of the following methods to identify an action for the AddrSymRecords collection to perform. Although written for the RSLogix 5 software product, the short examples following each method may be easily adapted to RSLogix 500. For example, type definitions may vary between products, and those differences must be considered when adapting code to the RSLogix 500 object model.

 Add	AddrSymRecord
<hr/>	
Use this method to create a new address/symbol record and add it to the AddrSymRecords collection.	

Syntax

Add() as AddrSymRecord

Returns

If successful the Address Symbol database record is created and added to the Address/Symbol Database Record collection. If unsuccessful, nothing is returned.

Example

The following code snippet makes the call to RSLogix to add an Address/Symbol database record to the Address/Symbol database record collection.

```
Set gAddrSymRecord = gAddrSymRecords.Add();
```



Duplicate

AddrSymRecord

Use this method to create a new record for an address/symbol with the record information of an address/symbol that is currently in the database. This is used as a shortcut to assign similar properties without having to retype information.

Important

If your source string is an address your new string must be an address. If your source string is a symbol, your new string must be a symbol.

Syntax

`Duplicate(SourceAddressOrSymbol as String, Scope as Integer, NewAddressOrSymbol as String) As AddrSymRecord`

Arguments

SourceAddressOrSymbol - The String used to identify the source address/symbol record.

Scope - The program file number that is local to the record. 0 is global

NewAddressOrSymbol - The String used to identify the new address/symbol record.

Returns

If successful, the duplicate record is returned. If unsuccessful Nothing is returned.

Example

The following code snippet shows both a valid and an invalid address/symbol record duplication request.

```
gAddrSymRecord = gAddrSymRecords.Duplicate("B3:0", "B3:1")
gAddrSymRecord = gAddrSymRecords.Duplicate("B3:0", "TEST")
'The line above will return Nothing because of mixed
'parameters (See Important Note in Duplicate description.)
```



GetRecordIndexViaAddrOrSym Long

Use this method to return the current Address/Symbol database record collection index indicated by either the address or symbol, and scope.

Syntax

`GetRecordIndexViaAddrOrSym (AddressOrSymbol as String, Scope as Integer) as Long`

Arguments

AddressOrSymbol - The String used to identify the source address/symbol record.

Scope - An integer that represents the file that is local to the symbol of the AddrSymRecord. A scope of 0 indicates that the symbol is global. If AddressOrSymbol is an address, use 0 for the scope.

Returns

If successful the index of the Address/Symbol database collection record is returned. If unsuccessful (-1) is returned.

Example

The following code snippet returns the index of the address B3:0.

```
Dim Index As Long
Index = gAddrSymRecords.GetRecordIndexViaAddrOrSym ("B3:0",0)
```



GetRecordViaAddrOrSym AddrSymRecord

Use this method to return the current Address/Symbol database record indicated by either the address, symbol, and the scope.

Syntax

GetRecordViaAddrOrSym (*AddressOrSymbol as String, Scope as Integer*) as AddrSymRecord

Arguments

AddressOrSymbol - The string that contains the address or the symbol of the Address/Symbol database record that is to be retrieved.

Scope - The program file number that is local to the record. 0 is global. If AddressOrSymbol is an address, use 0 for the scope.

Returns

If successful, the indicated Address/Symbol database collection record is returned. If unsuccessful Nothing is returned

Example

The following code snippet returns the Address/Symbol database record for B3:0.

```
gAddrSymRecord = gAddrSymRecords.GetRecordViaAddrOrSym ("B3:0",0)
```



GetRecordViaDesc AddrSymRecord

Use this method to return the next Address/Symbol database record whose description contains the search string.

Syntax

GetRecordViaDesc(StartingIndex as Long, DescriptionSearchString as String, CaseSensitive as Boolean, Wrap as Boolean) as AddrSymRecord

Arguments

StartingIndex - The zero-based index to start the search from. If the Address/Symbol database had 100 records, 0-99 would be the legal range for the starting index. This argument is passed by reference – you must specify it as a Long, not as an immediate.

DescriptionSearchString - The string that will be searched for in the Address/Symbol database record descriptions.

CaseSensitive - If set to True, the case of any letters in the DescriptionSearchString will be used to filter the search.

Wrap - If set to True, the search wraps past the last index of the database and will continue the search from index 0 to StartingIndex -1.

Returns

If successful, the indicated Address/Symbol database collection record is returned; if unsuccessful Nothing is returned. The StartingIndex will return the index of the AddrSymRecord that was found.

Example

The following call will perform a case-sensitive search from record 10 for a description that contains PLC-5. If the search reaches the end of the database, the search stops. It will not wrap back to record 0.

```
Dim Index As Long
Index = 10
gAddrSymRecord = gAddrSymRecords.GetRecordViaDesc(Index, "PLC-5",
True, False)
```



GetRecordViaIndex

AddrSymRecord

Use this method to return the current Address/Symbol database record indicated by the zero-based index.

Syntax

GetRecordViaIndex(Index as Long) As AddrSymRecord

Arguments

Index - The zero-based index that contains the address or the symbol of the Address/Symbol database record that is to be retrieved. If the Address/Symbol database had 100 records, 0-99 would be the legal range for the index. The number of records in the database can be found with the count property.

Returns

If successful, the indicated Address/Symbol database collection record is returned. If unsuccessful Nothing is returned.

Example

The following code snippet returns the Address/Symbol database record by providing the index identified as 123.

```
gAddrSymRecord = gAddrSymRecords.GetRecordViaIndex(123)
```



RemoveRecordViaAddrOrSym Boolean

Use this method to remove the current Address/Symbol database record indicated by either the address, symbol, and the scope.

Syntax

RemoveRecordViaAddrOrSym(*AddressOrSymbol as String, Scope as Integer*) as Boolean

Arguments

AddressOrSymbol - The string that contains the address or the symbol of the Address/Symbol database record that is to be retrieved.

Scope - The program file number that is local to the record. 0 is global. If your AddressOrSymbol string is an address, use a scope of 0.

Returns

If successful, the indicated Address/Symbol database collection record is removed and True is returned. If unsuccessful False is returned

Example

The following code snippet removes the Address/Symbol database record for B3:0.

```
Dim Result As Boolean
```

```
Result = gAddrSymRecords.RemoveRecordViaAddrOrSym ("B3:0",0)
```



RemoveRecordViaIndex Boolean

Use this method to remove the current Address/Symbol database record indicated by zero-based index.

Syntax

RemoveRecordViaIndex(*Index as Long*) As Boolean

Arguments

Index - The zero-based index that contains the address or the symbol of the Address/Symbol database record that is to be retrieved. If the Address/Symbol database had 100 records, 0-99 would be the legal range for the index. The number of records in the database can be found with the count property.

Returns

If successful, the indicated Address/Symbol database collection record is removed and a value of True is returned. If unsuccessful False is returned.

Example

The following code snippet removes the Address/Symbol database record by providing the index identified as 123.

```
Dim Result as Boolean
```

```
Result = gAddrSymRecords.RemoveRecordViaIndex(123)
```



SearchAndReplaceDesc Long

Use this method to replace text in the description of the next Address/Symbol database record whose descriptions contains the search string.

Syntax

```
SearchAndReplaceDesc(StartingIndex as Long, DescriptionSearchString as String,  
DescriptionReplaceString as String, CaseSensitive as Boolean, Wrap as Boolean,  
ReplaceAll as Boolean) as Long
```

Arguments

StartingIndex - The zero based index to start the search from. If the Address/Symbol database had 100 records, 0-99 would be the legal range for the starting index. This argument is passed by reference – you must specify it as a Long, not as an immediate.

DescriptionSearchString - The string that will be searched for in the Address/Symbol database record descriptions.

DescriptionReplaceString - The string that will replace any description search strings that are located.

CaseSensitive - If set to True, the case of any letters in the DescriptionSearchString will be used to filter the search.

Wrap - If set to True the search wraps past the last index of the database and continues from the beginning until a match is found or the current record's index matches the starting index. This parameter is ignored if ReplaceAll is True.

ReplaceAll- If set to True all of the instances of the DescriptionSearchString will be replaced throughout all of the descriptions in the Address/Symbol database. If ReplaceAll is set, the Wrap parameter is ignored.

Returns

The number of Address/Symbol database collection record descriptions that were changed is returned. The StartingIndex will return the index of the last AddrSymRecord where a replace had occurred.

Example

The following call performs a non case-sensitive search and replace from record 10 for a description that contains “test” and replace “test” with “Debug.” Since ReplaceAll is not True there will be only one replacement if there are any. If the search reaches the end of the database, the search will wrap back to record 0 and continue searching until either a match is found or the StartingIndex is reached.

```
Dim Index as Long
Index = 10
NumberOfReplacedDescriptions As Long
NumberOfReplacedDescriptions = gAddrSymRecords. SearchAndReplaceDesc
(Index, "test", "Debug", False, True, False)
```

Events

No events have been defined for the AddrSymRecords collection.

AddrSymRecord object

The AddrSymRecord object represents an Address/Symbol database record in the RSLogix project. Use it to return or set a value in any field. The AddrSymRecord object is obtained from the AddrSymRecords collection via the Add, GetRecord, and GetRecordViaDesc methods. You cannot create a new instance of a AddrSymRecord object with the CreateObject function.

Properties	Methods	Events
Above	SetAbove	-None-
Address	SetAddress	
Application	SetBelow	
Below	SetDescription	
Description	SetDeviceCode	
DeviceCode	SetScope	
Scope	SetSymbol	
Symbol	SetSymGroup	
SymbolGroup		

The following commented code example illustrates how you might access the AddrSymRecord object.







```

Private Sub Form_Load()
Set gAddrSymRecords = gProject.AddrSymRecords
Set gAddrSymRecord = gAddrSymRecords.Add()
If gAddrSymRecord Is Nothing Then
'if the AddrSymRecord object does not exist then display an error
MsgBox "Error getting Address Symbol record"
End If

```

Properties

In most cases properties are characteristics or attributes of an object. Using a property returns information about the object or causes a quality of the object to change. The following properties define the AddrSymRecord object.

 Above	String - Read Only
This property returns a text string (the Above string) from the AddrSymRecord. The Above string appears above an I/O point in a RSWire schematic diagram...	
 Address	String - Read Only
This property returns a string that identifies the address of the AddrSymRecord.	
 Application	Application - Read Only
This property returns an Application object that represents the RSLogix application.	
 Below	String - Read Only
This property returns a text string (the Below string) from the AddrSymRecord. The Below string appears below an I/O point in a RSWire schematic diagram.	
 Description	String - Read Only
This property returns a string that identifies the description in the AddrSymRecord.	
 DeviceCode	String - Read Only
This property returns a string that identifies the device code in the AddrSymRecord. Device codes correspond to device names in the PLC/SLC database and to device drawings in the RSWire I/O Builder database. Only I/O addresses may have device codes.	



Scope**Long - Read Only**

This property returns a long value that identifies the scope of the AddrSymRecord. 0 is global. Numbers 1-1999 represent the local program file number.



Symbol**String - Read Only**

This property returns a string that identifies the symbol of the AddrSymRecord.



SymbolGroup**String - Read Only**

This property returns a string that identifies name of the symbol group of the AddrSymRecord. An empty string indicates that the AddrSymRecord is not the member of a symbol group.

Methods

Using a method causes something to happen to an object. In most cases methods are actions. Use any of the following methods to identify an action for the AddrSymRecord object to perform. Although written for the RSLogix 5 software product, the short examples following each method may be easily adapted to RSLogix 500. For example, type definitions may vary between products, and those differences must be considered when adapting code to the RSLogix 500 object model.



SetAbove**Boolean**

Use this method to set the Above field of an AddrSymRecord. RSLogix uses this string to export to RSWire I/O Builder where it is placed above the device drawing on the resulting I/O schematic generated by RSWire.

Syntax

SetAbove(*AboveString As String*) as Boolean

Arguments

AboveString - The ASCII string (up to 9 characters) that will be used to set the Above field of the AddrSymRecord.

Returns

If successful the Above string is added to the AddrSymRecord and True is returned. If unsuccessful False is returned. The device code must be set before this member will work successfully.

Example

The following code snippet makes the call to set the Above field of the AddrSymRecord.

```
Result As Boolean
Result = gAddrSymRecord.SetAbove("abovetext")
```



SetAddress

Boolean

Use this method to set the address of an AddrSymRecord.

Syntax

SetAddress(*Address as String*) as Boolean

Arguments

Address - The ASCII string that identifies the address that the AddrSymRecord will be set to.

Returns

If successful True is returned; if unsuccessful False is returned.

Example

The following code snippet makes the call to RSLogix to set the address of AddrSymRecord.

```
Result As Boolean
Result = gAddrSymRecord.SetAddress("B3:0")
```



SetBelow

Boolean

Use this method to set the Below field of an AddrSymRecord. RSLogix uses this string to export to RSWire I/O Builder where it is placed below the device drawing on the resulting I/O schematic generated by RSWire.

Syntax

SetBelow(*BelowString As String*) as Boolean

Arguments

BelowString - The ASCII string (up to 9 characters) that will be used to set the Below field of the AddrSymRecord.

Returns

If successful the Below string is added to the AddrSymRecord and True is returned. If unsuccessful False is returned. The device code must be set before this member will return successfully.

Example

The following code snippet makes the call to set the Below field of the AddrSymRecord to "Test."

```
Result As Boolean
Result = gAddrSymRecord.SetBelow("TEST")
```



SetDescription

Boolean

Use this method to set the description field of the AddrSymRecord. All instructions having the same address will automatically have the same description.

Syntax

SetDescription(*Description as String*) as Boolean

Arguments

Description - The ASCII string that identifies the description to set for the AddrSymRecord. The length of this string is limited to the MaxDescriptionLineLength property of the Application Object.

Returns

If successful the description string is added to the AddrSymRecord and True is returned; if unsuccessful False is returned.

Example

The following code snippet makes the call to set the description of the AddrSymRecord to "Test Description."

```
Result As Boolean
Result = gAddrSymRecord.SetDescription("Test Description")
```



SetDeviceCode

Boolean

Use this method to set the device code of an AddrSymRecord. The device code is used in wiring diagrams created in the RSWire software package.

Syntax

SetDeviceCode(*DeviceCode as String*) as Boolean

Arguments

DeviceCode - The device code that represents the I/O point device code type.

Note: this is only available to AddrSymRecord types that have an input or output address. Any 11 character string will be accepted.

A valid list of device code strings for inputs and outputs follows.

Input Device Codes

2KNCL	3SSNCL	MMPNCA	PBNCO
2KNCM	3SSNCM	MMPNCG	PBNCR
2KNCR	3SSNCR	MMPNCR	PBNCY
2KNOL	3SSNOL	MMPNOA	PBNOB
2KNOM	3SSNOM	MMPNOG	PBNOE
2KNOR	3SSNOR	MMPNOR	PBNOG
2SSNCL	CRNC	MOPNCA	PBNOK
2SSNCM	CRNO	MOPNCG	PBNOO
2SSNCR	FSNC	MOPNCR	PBNOR
2SSNOL	FSNO	MOPNOA	PBNOR
2SSNOM	FTSMNC	MOPNOG	PBNOY
2SSNOR	FTSMNO	MOPNOR	PENC
3KNCC	FTSSNC	NCTC	PENO
3KNCL	FTSSNO	NCTO	PRXNC
3KNCM	LSHC	NOTC	PRXNO
3KNCR	LSHO	NOTO	PSNC
3KNOC	LSNC	PBNCB	PSNO
3KNOL	LSNO	PBNCE	SPIN
3KNOM	MCRNC	PBNCG	TSNC
3KNOR	MCRNO	PBNCK	TSNO

Output Device Codes

CNCOIL	PLTG	SPOT
CRCOIL	PLTR	SSOL
DSOIL	PLTW	TMCOIL
MTPLTA	PTPLTA	TRCOIL
MTPLTB	PTPLTB	TSOL
MTPLTC	PTPLTC	WBA
MTPLTG	PTPLTG	WBB
MTPLTR	PTPLTR	WBG
MTPLTW	PTPLTW	WBL
PLTA	RCOIL	WBR
PLTB	SIZEO	WBZ
PLTC	SIZER	

Returns

If successful the device code is set and a value of True is returned; if unsuccessful False is returned.

Example

The following code snippet makes the call to RSLogix to set the output device code for the AddrSymRecord to “SPOT.”

```
Result As Boolean
Result = gAddrSymRecord.SetDeviceCode("SPOT")
```



SetScope

Boolean

Use this method to set the local program file of the AddrSymRecord.

Syntax

SetScope(*Scope as Long*) as Boolean

Arguments

Scope - A Long that represents the file that is local to the symbol of the AddrSymRecord. A scope of 0 indicates that the symbol is global.

Returns

If successful True is returned. If unsuccessful False is returned.

Example

The following code snippet makes the call to set the scope of AddrSymRecord.

```
Result As Boolean
Result = gAddrSymRecord.SetScope(0) 'Makes the gAddrSymRecord global
```



SetSymbol

Boolean

Use this method to set the symbol of an AddrSymRecord.

Syntax

SetSymbol(*Symbol as String*) as Boolean

Arguments

Symbol - The string that contains the symbol that the AddrSymRecord is set to. This string length is limited by the MaxSymbolLength property of the Application Object.

Returns

If successful True is returned. If unsuccessful False is returned.

Example

The following code snippet makes the call to set the symbol of AddrSymRecord.

```
Result As Boolean
Result = gAddrSymRecord.SetSymbol("TEST")
```



SetSymGroup

Boolean

Use this method to set the symbol group of the AddrSymRecord. If the symbol group does not exist when this function is called, the symbol group is created.

Syntax

SetSymGroup(*SymGroup as String*) as Boolean

Arguments

SymGroup - The string that represents the name of the symbol group to which the symbol property will be added.

Returns

If successful True is returned. If unsuccessful False is returned.

Example

The following code snippet makes the call to set the symbol group for AddrSymRecord.

```
Result As Boolean
```

```
Result = gAddrSymRecord.SetSymGroup("TEST_SYM_GROUP")
```

```
'The Result should be True if in this project and if 'gAddrSymRecord  
has a valid address and symbol.
```

Events

No events have been defined for the AddrSymRecord object.

RungCmntPageTitleRecords collection

The RungCmntPageTitleRecords collection represents the collection of Rung Comment/Page Title database records (RungCommentPageTitleRecord) in the RSLogix project. The RungCommentPageTitleRecords collection can be obtained using the RungCommentPageTitleRecords property of the LogixProject object. The RungCommentPageTitleRecords collection is not creatable with the CreateObject function.

Properties	Methods	Events
Application	AddRecordAttachedProgFileAndRung	-None-
Count	AddRecordAttachedToAddress	
	DuplicateViaAddress	
	DuplicateViaFileRung	
	GetRecordViaAddress	
	GetRecordViaFileRung	
	GetRecordViaIndex	
	GetRecordViaPageTitle	
	GetRecordViaRungComment	
	RemoveRecordViaAddress	
	RemoveRecordViaFileRung	
	RemoveRecordViaIndex	
	SearchAndReplacePageTitle	
	SearchAndReplaceRungComment	

The following commented code example illustrates how you might get the RungCmntPageTitleRecords collection from the LogixProject object. This example adds error checking and notification.

```

'get RungCmntPageTitleRecords collection from LogixProject object
Set RungCmntPageTitleRecords =
gLogixProject.RungCmntPageTitleRecords
'if Logix failed to get the RungCmntPageTitleRecords collection then
'display an error and exit
If RungCmntPageTitleRecords Is Nothing Then
    MsgBox "ERROR: Could not get Rung Comment/Page Title Records!",
    vbExclamation, "ERROR"
    Exit Function
End If

```

Properties

In most cases properties are characteristics or attributes of an object. Using a property returns information about the object or causes a quality of the object to change. The following properties define the RungCmntPageTitleRecords collection.

 **Application** **Read Only**

This property returns an Application object that represents the RSLogix application.

 **Count** **Read Only**

This property returns a long value that represents the number of RungCmntPageTitle records saved with the project.

Methods

Using a method causes something to happen to an object. In most cases methods are actions. Use any of the following methods to identify an action for the RungCmntPageTitleRecords collection to perform. Although written for the RSLogix 5 software product, the short examples following each method may be easily adapted to RSLogix 500. For example, type definitions may vary between products, and those differences must be considered when adapting code to the RSLogix 500 object model.

 **AddRecordAttachedToProgFileAndRung** **RungCommentPageTitleRecord**

Use this method to create a new Rung Comment/Page Title Database Record, add it to the collection, and attach the record to a program file and rung.

Syntax

AddRecordAttachedToProgFileAndRung(*ProgFile as Long, Rung as Long*) as RungCommentPageTitleRecord

Arguments

ProgFile - The number of the program file where the documentation is to be attached.

Rung - The number of the rung to which the RungCommentPageTitleRecord is attached.

Returns

If successful the Rung Comment/Page Title database record is created and added to the Rung Comment/Page Title Database Record Collection.

Example

The following code snippet creates a new Rung Comment/Page Title in program file 3 attached to rung 12.

```
Set gRungCommentPageTitleRecord =  
gRungCommentPageTitleRecords.AddRecordAttachedToProgFileAndRung(3,12,)
```



AddRecordAttachedToAddress RungCommentPageTitleRecord

Use this method to create a new Rung Comment/Page Title Database Record, add it to the collection, and attach the record to an address.

Syntax

AddRecordAttachedToAddress(*Address as String*) as
RungCommentPageTitleRecord

Arguments

Address - The text string that contains the address where the documentation is to be attached.

Returns

If successful the Rung Comment/Page Title database record is created and added to the Rung Comment/Page Title Database Record collection. If unsuccessful, Nothing is returned.

Example

The following code snippet makes the call to RSLogix to add a Rung Comment/Page Title Database record to the Rung Comment/Page Title Database Record Collection.

```
Set gRungCommentPageTitleRecord =  
gRungCommentPageTitleRecords.AddRecordAttachedToAddress ("B3:0")
```



DuplicateViaAddress RungCommentPageTitleRecord

Use this method to create a new record for a Rung Comment/Page Title by providing a source address that is currently in the database from which the record information of a Rung Comment/Page Title can be duplicated and applied to a new address.

Syntax

DuplicateViaAddress(*SourceAddress* as String, *NewAddress* as String) As RungCommentPageTitleRecord

Arguments

SourceAddress - The string containing the address of the record to be duplicated.

NewAddress - The string containing the address of the duplicated record that will be returned if the method is successful.

Returns

If successful, the duplicate record is returned; if unsuccessful Nothing is returned.

Example

The following code snippet returns a Rung Comment/Page Title record provided that a record for B3:0 exists and that a record for B3:1 does not exist prior to the call of the Duplicate method.

```
gRungCommentPageTitleRecord =  
gRungCommentPageTitleRecords.DuplicateViaAddress("B3:0", "B3:1")
```



DuplicateViaFileRung RungCommentPageTitleRecord

Use this method to create a new record for a Rung Comment/Page Title by providing a file and rung number currently in the database from which the record information of a Rung Comment/Page Title can be duplicated and applied to a new rung.

Syntax

DuplicateViaFileRung(*SourceFileNumber* as Long, *SourceRungNumber* as Long, *DestFileNumber* as Long, *DestRungNumber* as Long) as RungCommentPageTitleRecord

Arguments

SourceFileNumber - The program file that contains the rung that is attached to the source record.

SourceRungNumber - The rung number that is attached to the source record.

DestFileNumber - The program file that contains the rung where the duplicated record will be created.

DestRungNumber - The rung number that is attached to the new (duplicate) destination record.

Returns

If successful the duplicate record is returned; if unsuccessful Nothing is returned.

Example

This call returns a Rung Comment/Page Title record provided that a record for program file 2, rung 0 exists and that a record for program file 4, rung 15 does not exist prior to the call of the Duplicate method.

```
gRungCommentPageTitleRecord =  
gRungCommentPageTitleRecords.DuplicateViaFileRung (2,0, "4,15")
```

GetRecordViaAddress RungCommentPageTitleRecord

Use this method to return the current Rung Comment/Page Title record indicated by the address.

Syntax

GetRecordViaAddress(*Address as String*) as RungCommentPageTitleRecord

Arguments

Address - The string that contains the address of the Rung Comment/Page Title record that is to be retrieved.

Returns

If successful, the indicated Rung Comment/Page Title record is returned; if unsuccessful Nothing is returned.

Example

This call returns a Rung Comment/Page Title provided that an existing record is attached to the address T4:0.

```
gRungCommentPageTitleRecord =  
gRungCommentPageTitleRecords.GetRecordViaAddress ("T4:0")
```

GetRecordViaFileRung RungCommentPageTitleRecord

Use this method to return the current Rung Comment/Page Title record indicated by the program file number and rung number.

Syntax

GetRecordViaFileRung(*FileNumber as Long, RungNumber as Long*) as RungCommentPageTitleRecord

Arguments

FileNumber - The number of the program file that contains the rung that is attached to the desired record.

RungNumber - The number of the rung that is attached to the desired record.

Returns

If successful the indicated Rung Comment/Page Title record is returned; if unsuccessful Nothing is returned.

Example

This call returns a Rung Comment/Page Title provided that an existing record is attached to program file 3, rung 4.

```
gRungCommentPageTitleRecord =  
gRungCommentPageTitleRecords.GetRecordViaFileRung(3,4)
```



GetRecordViaIndex

RungCommentPageTitleRecord

Use this method to return the current Rung Comment/Page Title record indicated by the zero based index.

Syntax

GetRecordViaIndex(*Index as Long*) as RungCommentPageTitleRecord

Arguments

Index - The zero-based index that contains the rung comment or the page title of the record that is to be retrieved. If the Rung Comment/Page Title database had 100 records, 0-99 would be the legal range for the index.

Returns

If successful, the indicated Rung Comment/Page Title record is returned; if unsuccessful Nothing is returned.

Example

This call returns a Rung Comment/Page Title record provided that there are at least 13 records in the RungCommentPageTitleRecord collection.

```
gRungCommentPageTitleRecord =  
gRungCommentPageTitleRecords.GetRecord(12)
```



GetRecordViaPageTitle RungCommentPageTitleRecord

Use this method to return the next Rung Comment/Page Title record whose page title contains the search string.

Syntax

GetRecordViaPageTitle(*Index as Long, PageTitleSearchString as String, CaseSensitive as Boolean, Wrap as Boolean*) as RungCommentPageTitleRecord

Arguments

Index - The zero-based index to start the search from. If the Rung Comment/Page Title database had 100 records, 0-99 would be the legal range for the index. This argument is passed by reference – you must specify it as a Long, not as an immediate.

PageTitleSearchString - The string that will be searched for in the page titles database.

CaseSensitive - If set to True, the case of any letters in the SearchString will be used to filter the search.

Wrap - If set to True a search wraps past the last index of the database and continues from the beginning until a match is found or the current record's index matches the starting index.

Returns

If successful the page title of the Rung Comment/Page Title record that contains the search string is returned; if unsuccessful Nothing is returned. The Index parameter will return the index of the RungCommentPageTitleRecord that was found.

Example

The following call performs a non case-sensitive search from record 10 for a page title that contains Page Title text. If the search reaches the end of the database, the search continues from 0 up to the starting index.

```
Dim Index As Long
Index = 10
gRungCommentPageTitleRecord =
gRungCommentPageTitleRecords.GetRecordViaPageTitle(Index, "Page Title
text", False, True)
```



GetRecordViaRungComment RungCommentPageTitleRecord

Use this method to return the next Rung Comment/Page Title record whose rung comment contains the search string.

Syntax

GetRecordViaRungComment(*Index as Long, RungCommentSearchString as String, CaseSensitive as Boolean, Wrap as Boolean*) as RungCommentPageTitleRecord

Arguments

Index - The zero-based index to start the search from. If the Rung Comment/Page Title database had 100 records, 0-99 would be the legal range for the starting index. This argument must be passed by reference as a Long.

RungCommentSearchString - The string searched for in the Rung Comment database.

CaseSensitive - If set to True, the case of any letters in the RungCommentsSearchString will be used to filter the search.

Wrap - If set to True the search wraps past the last index of the database and continues from the beginning until a match is found or the current record's index matches the starting index.

Returns

If successful the rung comment for the Rung Comment/Page Title record that contains the search string is returned; if unsuccessful Nothing is returned. The starting index will return the index of the last RungCommentPageTitleRecord that was found.

Example

The following call performs a non case-sensitive search from record 10 for a page title that contains Page Title text. If the search reaches the end of the database, the search continues from 0 up to the starting index.

```
Dim Index As Long
Index = 10

gRungCommentPageTitleRecord =
gRungCommentPageTitleRecords.GetRecordViaRungComment(Index, "PLC-5",
False, True)
```



RemoveRecordViaAddress Boolean

Use this method to remove a record from the Rung Comment/Page Title record by indicating its address.

Syntax

RemoveRecordViaAddress(*Address as String*) as Boolean

Arguments

Address - The string that contains the address of the Rung Comment/Page Title record that is to be removed.

Returns

If successful the indicated record is removed from the Rung Comment/Page Title collection and a value of True is returned; if unsuccessful False is returned.

Example

The following call removes the Rung Comment/Page Title record attached to address B3:0.

```
Dim Res As Boolean
Res = gRungCommentPageTitleRecords.RemoveRecordViaAddress("B3:0");
```

RemoveRecordViaFileRung Boolean

Use this method to remove a record from the Rung Comment/Page Title record by indicating the file number and rung number of the record.

Syntax

RemoveRecordViaFileRung(*FileNumber as Long, RungNumber as Long*) as Boolean

Arguments

FileNumber - The program file number that contains the rung to which the documentation is attached.

RungNumber - The rung number to which the documentation is attached.

Returns

If successful, the indicated record is removed from the Rung Comment/Page Title collection and a value of True is returned; if unsuccessful False is returned.

Example

The following call removes the Rung Comment/Page Title record attached to ProgFile 4, rung 2 from the Rung Comment/Page Title database.

```
Dim Res As Boolean
Res = gRungCommentPageTitleRecords.RemoveRecordViaFileRung(4,2)
```

RemoveRecordViaIndex Boolean

Use this method to remove the Rung Comment/Page Title record indicated by the zero based index.

Syntax

RemoveRecordViaIndex(*Index as Long*) as Boolean

Arguments

Index - The zero based index that contains the rung comment or the page title of the record that is to be removed. If the Rung Comment/Page Title database had 100 records, 0-99 would be the legal range for the index.

Returns

If successful, the indicated record is removed from the Rung Comment/Page Title collection and a value of True is returned; if unsuccessful False is returned.

Example

This example removes record 2 (the third record) from the zero-based index.

```
Dim Res As Boolean
```

```
Res = gRungCommentPageTitleRecords.RemoveRecordViaIndex(2)
```



SearchAndReplacePageTitle Long

Use this method to replace text in the page title of the next Rung Comment/Page Title record whose page title contains the search string.

Syntax

SearchAndReplacePageTitle(Index as Long, PageTitleSearchString as String, PageTitleReplaceString as String, CaseSensitive as Boolean, Wrap as Boolean, ReplaceAll as Boolean) as Long

Arguments

Index - The zero-based index to start the search from. If the Rung Comment/Page Title database had 100 records, 0-99 would be the legal range for the index. This argument is passed by reference – you must specify it as a Long, not as an immediate.

PageTitleSearchString - The string that is searched for in the Page Title database.

PageTitleReplaceString - The replacement string to be used in place of the SearchString page title database.

CaseSensitive - If set to True, the case of any letters in the SearchString will be used to filter the search.

Wrap - If set to True a search wraps past the last index of the database and continues from the beginning until a match is found or the current record's index matches the starting index.

ReplaceAll - If set to True all instances of the SearchString will be replaced throughout all of the rung comments in the Rung Comment/Page Title database. If ReplaceAll is set, the Wrap parameter is ignored.

Returns

The number of Rung Comment/Page Title database record page titles that were changed is returned. Index will contain the index of the last changed record if the number of changes is greater than 0.

Example

The following call will perform a non case-sensitive search and replace from record 10 for a page title that contains “test” and replace “test” with “debug.” Since ReplaceAll is not True there will only be one replacement if there are any. If the search reaches the end of the database, the search will wrap back to record 0 and continue searching until either a match is found or the Index is reached.

```
Dim Index As Long
Index = 10
CommentsReplaced As Long
CommentsReplaced = gRungCommentPageTitleRecords.
SearchAndReplacePageTitle (Index, "test", "Debug", False, True,
False)
```



SearchAndReplaceRungComment Long

Use this method to replace text in the rung comment of the next Rung Comment/Page Title record whose rung comment contains the search string.

Syntax

SearchAndReplaceRungComment(*Index as Long, RungCommentSearchString as String, RungCommentReplaceString as String, CaseSensitive as Boolean, Wrap as Boolean, ReplaceAll as Boolean*) as Long

Arguments

Index - The zero-based index to start the search from. If the Rung Comment/Page Title database had 100 records, 0-99 would be the legal range for the index. This argument is passed by reference – you must specify it as a Long, not as an immediate.

PageTitleSearchString - The string that is searched for in the Rung Comment database.

PageTitleReplaceString - The replacement string to be used in place of the SearchString rung comment database.

CaseSensitive - If set to True, the case of any letters in the SearchString will be used to filter the search.

Wrap - If set to True a search wraps past the last index of the database and continues from the beginning until a match is found or the current record's index matches the starting index.

ReplaceAll - If set to True all instances of the SearchString will be replaced throughout all of the rung comments in the Rung Comment/Page Title database. If ReplaceAll is set, the Wrap parameter is ignored.

Returns

The number of Rung Comment/Page Title database record page titles that were changed is returned. Index will contain the index of the last changed record if the number of changes is greater than 0.

Example

The following call will perform a non case-sensitive search and replace from record 10 for a rung comment that contains "test" and replace "test" with "debug." Since ReplaceAll is not True there will only be one replacement if there are any. If the search reaches the end of the database, the search will wrap back to record 0 and continue searching until either a match is found or the Index is reached.

```
Dim Index As Long
Index = 10
CommentsReplaced As Long
CommentsReplaced = gRungCommentPageTitleRecords.
SearchAndReplaceRungComment (Index, "test", "Debug", False, True,
False)
```

Events

No events have been defined for the RungCmntPageTitleRecords collection.

RungCmntPageTitleRecord
object

The RungCmntPageTitleRecord object represents a Rung Comment/Page Title record in the RSLogix project. The RungCommentPageTitleRecord is obtained via the RungCommentPageTitleRecords collection Add, GetRecord, GetRecordViaRungComment, and GetRecordViaPageTitle member functions. RungCommentPageTitleRecord is not creatable with the CreateObject function.

Properties	Methods	Events
Address	SetAddress	-None-
Application	SetPageTitle	
IsAttachedToAddress	SetProgFileAndRung	
PageTitle	SetRungComment	
ProgFile		
RungComment		
RungNumber		

The following commented code example illustrates how you might access the RungCmntPageTitleRecord object.








```

Private Sub Form_Load()
Set gRungCmntPageTitleRecords = gLogixProject.RungCmntPageTitleRecords
Set gRungCmntPageTitleRecord =
gRungCmntPageTitleRecords.AddRecordAttachedtoAddress("B3:0")
If gRungCmntPageTitleRecord Is Nothing Then
'if the RungCmntPageTitleRecord object does not exist
'then display an error
MsgBox "Error getting Rung Comment Page Title record"
End If

```

Properties

In most cases properties are characteristics or attributes of an object. Using a property returns information about the object or causes a quality of the object to change. The following properties define the RungCmntPageTitleRecord object.

 Address	String - Read Only
This property returns a string containing the address of the RungCommentPageTitleRecord. This string will be empty if the record is attached to a program file/rung number combination.	
 Application	Application - Read Only
This property returns an Application object that represents the RSLogix application.	
 IsAttachedToAddress	Boolean - Read Only
If this property is True the record is attached to an address. If this property is False the record is attached to a program file/rung number combination.	
 PageTitle	String - Read Only
This property returns a string containing the page title of the RungCommentPageTitleRecord.	
 ProgFile	Long - Read Only
This property returns a long containing the program file number of the RungCommentPageTitleRecord. This property will return (-1) if the record is attached to an address.	
 RungComment	String - Read Only
This property returns a string containing the rung comment of the RungCommentPageTitleRecord.	
 RungNumber	Long - Read Only
This property returns a long containing the rung number of the RungCommentPageTitleRecord. This property will return (-1) if the record is attached to an address.	

Methods

Using a method causes something to happen to an object. In most cases methods are actions. Use any of the following methods to identify an action for the RungCmntPageTitleRecord object to perform. Although written for the RSLogix 5 software product, the short examples following each method may be easily adapted to RSLogix 500. For example, type definitions may vary between products, and those differences must be considered when adapting code to the RSLogix 500 object model.

 SetAddress	Boolean
---	----------------

Use this method to set the Address of a RungCommentPageTitleRecord. If successful this method will set the AttachedToAddress property True.

Syntax

SetAddress(*Address as String*) as Boolean

Arguments

Address - The string that contains the address that is attached to the RungCommentPageTitleRecord.

Returns

If successful True is returned; if unsuccessful False is returned.

Example

The following code snippet makes the call to RSLogix to set the address of RungCommentPageTitleRecord.

```
Result As Boolean
```

```
Result = gRungCommentPageTitleRecord.SetAddress("B3:0")
```

 SetPageTitle	Boolean
---	----------------

Use this method to set the page title text of a RungCommentPageTitleRecord.

Syntax

SetPageTitle(*PageTitle as String*) As Boolean

Arguments

PageTitle - The text string that contains the page title text of the RungCommentPageTitleRecord.

Returns

If successful True is returned; if unsuccessful False is returned.

Example

The following code snippet makes the call to RSLogix to set the page title text of RungCommentPageTitleRecord.

Result As Boolean

```
Result = gRungCommentPageTitleRecord.SetPageTitle("This section  
controls the Main Transfer Motor on Line 2")
```

SetProgFileAndRung Boolean

Use this method to set the program file and rung of a RungCommentPageTitleRecord. If successful this method sets the AttachedToAddress property False.

Syntax

SetProgFileAndRung(*ProgFile as Long, Rung as Long*) As Boolean

Arguments

ProgFile - The number of the program file that contains the rung that is attached to the RungCommentPageTitleRecord.

Rung - The number of the rung to which the RungCommentPageTitleRecord is attached.

Returns

If successful True is returned, if unsuccessful False is returned.

Example

The following code snippet makes the call to RSLogix to attach the Rung Comment/Page Title to program file 3 rung 2. The result will be successful provided that program file 3 is a ladder file that contains rung 2 and program file 3, rung 2 is not attached to another RungCommentPageTitle record.

Result As Boolean

```
Result = gRungCommentPageTitleRecord.SetProgFileAndRung(3,2)
```

SetRungComment Boolean

This method sets the rung comment of a RungCommentPageTitleRecord.

Syntax

SetRungComment(*RungComment as String*) as Boolean

Arguments

RungComment - The text string that contains the rung comment of the RungCommentPageTitleRecord.

Returns

If successful True is returned; if unsuccessful False is returned.

Example

The following code snippet makes the call to RSLogix to set the rung comment text of the RungCommentPageTitleRecord.

```
Result As Boolean
```

```
Result = gRungCommentPageTitleRecord.SetRungComment("This section  
controls the Main Transfer Motor on Line 2")
```

Events

No events have been defined for the RungCmntPageTitleRecord object.

17 PasswordPrivilegeConfig object

Note: The PasswordPrivilegeConfig object applies to RSLogix 5 only.

The PasswordPrivilegeConfig object represents the passwords and privileges configuration in the RSLogix project. The PasswordPrivilegeConfig is obtained using the PasswordPrivilegeConfig property of the LogixProject object. PasswordPrivilegeConfig is not creatable with the CreateObject function.

Properties	Methods	Events
Application	AddNodePrivilegeEntry	-None-
CurrentClass	ChangeNodePrivilegeInfo	
NodePrivilegeEntryCount	ClassLogin	
	DownloadPrivChanges	
	GetChannelPrivileges	
	GetDataFilePrivileges	
	GetDefaultClass	
	GetFeaturePrivileges	
	GetNodePrivilegeInfo	
	GetProgFilePrivileges	
	IsClassPasswordProtected	
	RefreshChannelPrivsFromOnline	
	RefreshPassPrivsFromOnline	
	RemoveNodePrivilegeEntry	
	SetChannelPrivileges	
	SetClassPassword	
	SetDataFilePrivileges	
	SetDefaultClass	
	SetFeaturePrivileges	
	SetProcessorPassword	
	SetProgFilePrivileges	




The following commented code example illustrates how you might access the PasswordPrivilegeConfig object.

```
Dim gPassPriv As RSLogix5.PasswordPrivilegeCfg

If LogixProject.Processor.HasPasswordPrivileges Then
    gPassPriv = LogixProject.PasswordPrivilegeCfg()
    If gPassPriv Is Nothing Then
        MsgBox "Error: Could not get password/privilege config!"
            vbExclamation , "Error"
        Exit Function
    End If
End If
```

Properties

In most cases properties are characteristics or attributes of an object. Using a property returns information about the object or causes a quality of the object to change. The following properties define the PasswordPrivilegeConfig object.

	Application	Application - Read Only
<hr/>		
This property returns an Application object that represents the RSLogix application.		
<hr/>		
	CurrentClass	Integer - Read Only
<hr/>		
This property returns an integer value that returns the current class that the project is logged into.		
<hr/>		
	NodePrivilegeEntryCount	Application - Read Only
<hr/>		
This property returns the number of active node privilege entries.		

Methods

Using a method causes something to happen to an object. In most cases methods are actions. Use any of the following methods to identify an action for the PasswordPrivilegeConfig object to perform.



AddNodePrivilegeEntry Short

Normally a station/node linked to a channel has the same privilege class as the channel it is linked to. You can, however, specify class privileges for a node separately. Node privileges override the default privilege class of the channel. Use this method to add a node privilege entry to the node privileges list in the current project.

Syntax

AddNodePrivilegeEntry (*Channel as lgxChannel, RemoteStation as Long, RemoteBridgeLinkID as Long, Class as Integer*) as Short

Arguments

Channel - The channel that is used to perform communications. The possible channels are lgxPLC5_Ch0, or any channel that the current processor could have configured for DH+ communications. The valid lgxChannel types are listed at GetDefaultClass on page 178.

RemoteStation - The station address of the node which will be placed in the privilege class specified in the Node Privilege table, rather than the class specified as the default class for the given channel. If the channel is lgxChan0 the remote station can only be 0. If the channel is lgxChan1A or lgxChan1B the remote station can be set within the range of 0-77 octal.

RemoteBridgeLinkID - When you are using DH+ networks through a PLC-5/250, the link number is used to identify the DH+ networks. You specify this link number on the PLC-5/250 configuration screens. If you are not using DH+ bridging, set this field to 0 to specify it as a local network. If you are using DH+ bridging, specify the link number of the network where the device establishing communications resides. The valid range is 0-65536.

Class - This is the privilege class which the node specified will be placed in once communications are established, rather than the default class assigned to the specified channel.

Returns

Returns (-1) if unsuccessful, otherwise the method will have been successful.

A (-1) is be returned if:

- the channel cannot be configured for DH+
- the RemoteStation is not an octal number (for example: 8)
- the RemoteStation or RemoteStationLinkID are outside the valid range
- the class currently logged into does not have “modify privilege” rights
- the specified channel had an existing node privilege entry

Example

The following code snippet makes the call to RSLogix to add a privilege node entry to the current project which is a 5/40 series B revision processor. This processor type has channel 2A which can be configured for DH+. We will configure the station to 16 and the link ID to 10 and make class 2 the default class for going online via channel 2A.

```
Index As Integer
```

```
Index = gPassPriv.AddNodePrivilegeEntry(lgxChan2A, 16, 10, 2)
```



ChangeNodePrivilegeInfo

Boolean

Use this method to change settings in a node privilege entry in the current project.

Syntax

```
ChangeNodePrivilegeInfo(Index as Long, Channel as lgxChannel,  
RemoteStation as Long, RemoteBridgeLinkID as Long, Class as Integer) as  
Boolean
```

Arguments

Index - The 0-based number representing the entry in the list of node privileges. If there are 10 entries in the list of node privileges, then the existing indices would range from 0-9.

Channel - The channel that is used to perform communications. The possible channels are lgxPLC5_Ch0 or any channel that the current processor could have configured for DH+ communications. The valid lgxChannel types are listed at GetDefaultClass on page 178.

RemoteStation - The station address of the node which will be placed in the privilege class specified in the Node Privilege table rather than the class specified as the default class for the given channel. If the channel is lgxPLC5_Ch0, the remote station can only be 0. If the channel is lgxPLC5_Ch1A or lgxPLC5_Ch1B, the remote station can be set within the range of 0-77 octal.

RemoteBridgeLinkID - When you are using DH+ networks through a PLC-5/250, the link number is used to identify the DH+ networks. You specify this link number on the PLC-5/250 configuration screens. If you are not using DH+ bridging, set this field to 0 to specify it as a local network. If you are using DH+ bridging, specify the link number of the network where the device establishing communications resides. The valid range is 0-65536.

Class - The privilege class which the node specified will be placed in when communications are established rather than the default class assigned to the specified channel.

Returns

Returns True if successful, otherwise returns False. False is returned if:

- the channel cannot be configured for DH+
- the RemoteStation is not an octal number (for example: 8) within the appropriate range
- the RemoteStation or RemoteStationLinkID are outside the valid range
- the Index is outside of the existing range
- the class currently logged into does not have “modify privilege” rights

Example

The following code snippet makes the call to RSLogix to change a privilege node entry in the current project which is a 5/40 series B revision B processor. The existing entry at Index 1 is configured using channel 2A, station to 16, linkID 10 and class 2 for the default class. The call in the example will change the settings to channel 1B, station 2, remote link ID 0 and class 3 as the default class for going online via channel 1B.

```
Result As Boolean  
Result = gPassPriv.ChangeNodePrivilegeInfo(1, lgxPLC5_Ch1B, 2, 0, 3)
```



ClassLogin

Boolean

Use this method to log into an offline class by passing the correct password for the class.

Syntax

ClassLogin(*Class as Integer, Password as String*) as Boolean

Arguments

Class - The class that the user wants to log into. The legal range is 1-4.

Password - The password for the class in which you are logging in.

Returns

True is returned if this method was successful, otherwise False is returned. False would be returned if the incorrect password was used for the specified class. If the class is not password protected, pass an empty string as the Password parameter.

Example

The following code snippet makes the call to RSLogix to log into class 2, providing the password 'password.'

Result As Boolean

```
Result = gPassPriv.ClassLogin(2, "password")
```



DownloadPrivChanges

Boolean

Use this method to download the password/privilege settings to the processor from the project while online. This method updates all of the processor's password/privilege settings except channel, data file, and program file privileges.

Syntax

DownloadPrivChanges() as Boolean

Returns

Returns True if successful, otherwise returns False.

Example

The following code snippet downloads the online processor's password/privilege settings.

bResult As Boolean

```
bResult = gPassPriv.DownloadPrivChanges()
```



GetChannelPrivileges

lgxPrivilege

Use this method to set the specified channel privileges in the current project. If the project is online use *RefreshChannelPrivsFromOnline* in order to ensure that the project matches the information stored in the online channel privilege image.

Syntax

GetChannelPrivileges(*Channel as lgxChannel, Class as Integer*) as lgxPrivilege

Arguments

Channel - The channel that contains the privilege class. Legal lgxChannel types include:

- (0) lgxPLC5_Ch0
- (1) lgxPLC5_Ch1A
- (2) lgxPLC5_Ch1B
- (3) lgxPLC5_Ch2A
- (4) lgxPLC5_Ch2B

- (5) lgxPLC5_Ch2
- (6) lgxPLC5_Ch3A
- (7) lgxPLC5_Offline

Class - The class number for which the privileges will be retrieved. The legal range is 1-4.

Returns

The method returns one of the following values.

- (0) lgxNoPriv
- (1) lgxReadPriv
- (2) lgxWritePriv
- (3) lgxReadWritePriv
- (4) lgxFailedToGetPriv

Example

The following code snippet makes the call to RSLogix to get the class 4 privileges for channel 3A.

```

Rights As lgxPrivilege
Rights = gPassPriv.GetChannelPrivileges(lgxPLC5_Ch3A, 4)
If Rights <> lgxFailedToGetPriv Then
    If (Rights = lgxNoPriv) Then
        MsgBox("No Privileges")
    ElseIf (Rights = lgxReadPriv)
        MsgBox("Read Privileges")
    ElseIf (Rights = lgxWritePriv) Then
        MsgBox("Write Privileges")
    ElseIf (Rights = lgxReadWritePriv)
        MsgBox("Read and Write Privileges")
    End If
End If

```



GetDataFilePrivileges lgxPrivilege

Use this method to get the data file privileges in the current project. If the project is online with the processor, the privileges for the specified file will be uploaded.

Syntax

GetDataFilePrivileges(*DataFile as Long, Class as Integer*) as lgxPrivilege

Arguments

DataFile - The number of the data file that you wish to retrieve class privileges for.

Class - The class number for which the privileges will be retrieved. The legal range is 1-4.

Returns

The method returns one of the following values.

- (0) lgxNoPriv
- (1) lgxReadPriv
- (2) lgxWritePriv
- (3) lgxReadWritePriv
- (4) lgxFailedToGetPriv

Example

The following code snippet makes the call to RSLogix to get the class 3 privileges for data file 5.

```
Rights As lgxPrivilege
Rights = gPassPriv.GetDataFilePrivileges(5, 3)
If Rights <> lgxFailedToGetPriv Then
    If (Rights = lgxNoPriv) Then
        MsgBox("No Privileges")
    ElseIf (Rights = lgxReadPriv)
        MsgBox("Read Privileges")
    ElseIf (Rights = lgxWritePriv) Then
        MsgBox("Write Privileges")
    ElseIf (Rights = lgxReadWritePriv)
        MsgBox("Read and Write Privileges")
    End If
End If
```



GetDefaultClass

Long

Use this method to get the default class of the offline editor or of any of the processor's channels.

Syntax

GetDefaultClass(*Channel as lgxChannel*) as Long

Arguments

Channel - The channel that contains the privilege class. The possible lgxChannel types include:

- (0) lgxPLC5_Ch0
- (1) lgxPLC5_Ch1A
- (2) lgxPLC5_Ch1B
- (3) lgxPLC5_Ch2A

- (4) lgxPLC5_Ch02B
- (5) lgxPLC5_Ch2
- (6) lgxPLC5_Ch3A
- (7) lgxPLC5_Offline

Returns

An integer is returned which represents the class number. 0 is returned if the channel does not exist in the processor.

Example

The following code snippet makes the call to RSLogix to get the default class for channel 0.

```
nClass As Long
nClass = gPassPriv.GetDefaultClass(lgxPLC5_CH0)
```

GetFeaturePrivileges lgxBinary

Use this method to get the specified feature privileges in the current project.

Syntax

GetFeaturePrivileges(*PrivilegeType as lgxPrivilege, Class as Integer*) as lgxBinary

Arguments

PrivilegeType - The possible lgxPrivilege types include:

- (0) lgxPrivModify
- (1) lgxPrivDataFileCreateDelete
- (2) lgxPrivProgFileCreateDelete
- (3) lgxPrivLogicalWrite
- (4) lgxPrivPhysicalWrite
- (5) lgxPrivLogicalRead
- (6) lgxPrivPhysicalRead
- (7) lgxPrivModeChange
- (8) lgxPrivIOForce
- (9) lgxPrivSFCForce
- (10) lgxPrivClearMemory
- (11) lgxPrivDownload
- (12) lgxPrivOnlineEdit
- (13) lgxPrivEditPassword

Class - The class from which the privilege status will be received.

Returns

If the method is successful, it returns `lgxEnabled` or `lgxDisabled`. Otherwise, it returns `lgxInvalid`.

Example

The following code snippet makes the call to `RSLogix` to get the online editing privileges for class 4.

```
Result As lgxBinary

Result = gPassPriv.GetFeaturePrivileges(lgxPrivOnlineEdit, 4)

If Result = lgxEnabled Then
    MsgBox("Class 4 has Online Edit Privileges")
ElseIf Result = lgxDisabled Then
    MsgBox("Class 4 does not have Online Edit Privileges")
Else
    MsgBox("Error getting Class 4 Edit Privileges")
End If
```



GetNodePrivilegeInfo Boolean

Use this method to get the specified node privilege in the current project.

Syntax

`GetNodePrivilegeInfo(Index as Long, Channel as lgxChannel, RemoteStation as Long, RemoteBridgeLinkID as Long, Class as Integer) as Boolean`

Arguments

Index - The 0-based number representing the entry in the list of node privileges. If there are 10 entries in the list of node privileges, then the existing indexes would range from 0-9.

Channel - This parameter will receive the `lgxChannel` that is specified by the entry located with the *Index*. Pass this parameter by reference.

RemoteStation - This parameter will receive the remote station number specified by the entry located with the *Index*. Pass this parameter by reference.

RemoteBridgeLinkID - This parameter will receive the remote bridge link ID specified by the entry located with the *Index*. Pass this parameter by reference.

Class - This parameter will receive the class specified by the entry located with the *Index*. Pass this parameter by reference.

Returns

Returns True if successful, otherwise returns False. If False is returned the Channel, RemoteStation, RemoteBridgeLinkID, and the Class parameters will not be updated by this method.

Example

The following code snippet makes the call to RSLogix to get the node privilege information from each node entry and sends some of the information to the user via a message box.

```
Chan As lgxChannel
ChannelString As String
StationNumber As Long
StationNumberString As String
BridgeLinkIDNumber As Long
class As Integer
msg As String
Count As Long
Index As Long
IndexString As String
Count = gPassPriv.GetNodePrivilegeEntryCount() - 1

For Index = 0 To Count

    If gPassPriv.GetNodePrivilegeInfo(Index, Chan,
    StationNumber, BridgeLinkIDNumber, class) Then

        ChannelString = Switch(Chan = lgxChan0, "Channel 0", Chan =
        lgxChan1A, "Channel DH+ 1A", Chan = lgxChan1B, "Channel DH+ 1B")

        StationNumberString = Format(StationNumber)

        IndexString = Format(Index)

        Msg = "The node privilege in entry "& IndexString "has" &
        ChannelString & " for the channel and "& StationNumberString
        & " for the Remote Station Number"

        MsgBox(Msg)
    End If

Next Index
```

 **GetProgFilePrivileges** **lgxPrivilege**

Use this method to get the specified program file privileges in the current project. If the project is online with the processor, the privileges for the specified file will be uploaded.

Syntax

GetProgFilePrivileges(*ProgFile as Long, Class as Integer*) as lgxPrivilege

Arguments

ProgFile - The number of the program file that you wish to retrieve class privileges for.

Class - The class number for which the privileges will be retrieved. The legal range is 1-4.

Returns

Returns lgxNoPriv, lgxReadPriv, lgxReadWritePriv, or lgxFailedToGetPriv.

Example

The following code snippet makes the call to RSLogix to get the class 3 privileges for program file 2.

Rights As lgxPrivilege

```
Rights = gPassPriv.GetProgFilePrivileges(2, 3)
If (Rights = lgxNoPriv) Then
    MsgBox("No Privileges")
ElseIf (Rights = lgxReadPriv)
    MsgBox("Read Privileges")
ElseIf (Rights = lgxWritePriv) Then
    MsgBox("Write Privileges")
ElseIf (Rights = lgxReadWritePriv)
    MsgBox("Read and Write Privileges")
End If
```

 **IsClassPasswordProtected** **Boolean**

Use this method to return if the login class has been password protected.

Syntax

IsClassPasswordProtected(*Class as Integer*) as Boolean

Arguments

Class - The class that is checked for a password.

Returns

If the class is protected, True is returned; otherwise False is returned.

Example

The following code snippet makes the call to RSLogix to check if class 1 is password protected.

```
Result As Boolean
Result = gPassPriv.IsClassPasswordProtected(1)
```

RefreshChannelPrivsFromOnline Boolean

Use this method to upload the channel privilege settings from the processor while online. This method will update the project's privilege settings. If the project is online using this method before calling *GetChannelPrivileges* will ensure that the channel privilege information in the project matches the channel privilege in the online image.

Syntax

RefreshChannelPrivsFromOnline() as Boolean

Returns

Returns True if successful, otherwise returns False.

Example

The following code snippet uploads the online processor's channel privilege settings.

```
bResult As Boolean
bResult = gPassPriv.RefreshChannelPrivsFromOnline()
```

RefreshPassPrivsFromOnline Boolean

Use this method to upload the password/privilege settings from the processor while online. This method will update all of the project's password/privilege settings except channel, data file, and program file privileges.

Syntax

RefreshPassPrivsFromOnline() as Boolean

Returns

Returns True if successful, otherwise False.

Example

The following code snippet uploads the online processor's password/privilege settings.

```
bResult As Boolean  
bResult = gPassPriv.RefreshPassPrivsFromOnline()
```

RemoveNodePrivilegeEntry **Boolean**

Use this method to remove a node privilege entry from the node privileges list in the current project.

Once an entry is removed, the indices of the existing entries above the removed entry are decreased by one.

For example: if the entry at index 0 is removed, the entry at index 1 becomes index 0, the entry index 2 becomes 1, etc.

Syntax

RemoveNodePrivilegeEntry(*Index as Short*) as Boolean

Arguments

Index - the 0-based number representing the entry in the list of node privileges. If there are 10 entries in the list of node privileges, then the existing indices would range from 0-9. An integer may be used for this parameter in Visual Basic since Visual Basic does not support shorts.

Returns

Returns True if successful, otherwise returns False. False is returned if the Index is outside of the index range of the currently existing node privilege entries.

Example

The following code snippet makes the call to RSLogix to remove a privilege node entry from the node privileges list. You could use *GetNodePrivilegeInfo* to get the privilege info settings to determine which entry to remove.

```
Result As Boolean  
Result = gPassPriv.RemoveNodePrivilegeEntry(0)
```

SetChannelPrivileges **Boolean**

Use this method to set the specified channel privileges in the current project.

Syntax

SetChannelPrivileges(*Channel as IgxChannel, Class as Integer, Privilege as IgxPrivilege*) as Boolean

Arguments

Channel - The channel that contains the privilege class. Legal lgxChannel types include:

- (0) lgxPLC5_Ch0
- (1) lgxPLC5_Ch1A
- (2) lgxPLC5_Ch1B
- (3) lgxPLC5_Ch2A
- (4) lgxPLC5_Ch2B
- (5) lgxPLC5_Ch2
- (6) lgxPLC5_Ch3A
- (7) lgxPLC5_Offline

Class - The class number for which the privileges will be retrieved. The legal range is 1-4.

Privilege - The privileges that are enabled for the specified class and channel. Valid lgxPrivilege types are:

- (0) lgxNoPriv
- (1) lgxReadPriv
- (2) lgxWritePriv
- (3) lgxReadWritePriv
- (4) lgxFailedToGetPriv

Returns

Returns True if successful, otherwise returns False.

Example

The following code snippet makes the call to RSLogix to set the class 3 privileges for channel 2 to Write only.

```
Rights As lgxPrivilege
Rights = lgxWritePriv
gPassPriv.SetChannelPrivileges(lgxPLC5_Ch2, 3, Rights)
```



SetClassPassword

Boolean

Use this method to set the password for the indicated class.

Syntax

SetClassPassword(*OldPassword as String, NewPassword as String, Class as Integer*) as Boolean

Arguments

OldPassword - The string that contains the old password. If there is no old password, use an empty string.

NewPassword - The string that contains the new password. The password is limited to 10 characters in length.

Class - The class that is checked for a password.

Returns

If the new password is set, True is returned; otherwise False is returned.

Example

The following code snippet makes the call to RSLogix to change the password of class 1.

```
Result As Boolean
```

```
Result = gPassPriv.SetClassPassword("oldpasswrd", "newpass", 1)
```



SetDataFilePrivileges Boolean

Use this method to set the specified data file privileges in the current project.

Syntax

SetDataFilePrivileges(*DataFile as Long, Class as Integer, Privilege as lgxPrivilege*) as Boolean.

Arguments

DataFile - The number of the data file that you wish to set class privileges for.

Class - The class number for which the privileges will be retrieved. The legal range is 1-4.

Privileges - The privileges that are enabled for the specified class and data file. Valid lgxPrivilege types are:

- (0) lgxNoPriv
- (1) lgxReadPriv
- (2) lgxWritePriv
- (3) lgxReadWritePriv
- (4) lgxFailedToGetPriv

Returns

Returns True if successful, otherwise returns False. False will be returned if the class logged it to does not have “modify privilege” rights.

Example

The following code snippet makes the call to RSLogix to set the class 2 privileges for data file 7 to disable read and write privileges.

```
Rights As lgxPrivilege
Rights = lgxNoPriv
gPassPriv.SetDataFilePrivileges(7, 2, Rights)
```



SetDefaultClass

Boolean

Use this method to set the default class of the offline editor or any of the processor's channels.

Syntax

SetDefaultClass(*Channel as lgxChannel, Class as Integer*) as Boolean

Arguments

Channel - The channel to assign the class to.

Class - The class that will be assigned to the channel.

Returns

If successful, True is returned, otherwise False is returned. This may be unsuccessful if *lgxClassError* is used for class, or if the channel does not exist in the processor, or if the class of the currently logged in user does not have the privilege to modify privileges.

Example

The following code snippet makes the call to RSLogix to set the default class for channel 0.

```
Result As Boolean
Result = gPassPriv.SetDefaultClass(lgxPLC5_Ch1A, 2)
```



SetFeaturePrivileges

Boolean

Use this method to set the specified feature privileges in the current project.

Syntax

SetFeaturePrivileges(*PrivilegeType as lgxPrivilege, Class as Integer, Enabled as Boolean*) as Boolean

Arguments

PrivilegeType - The possible `lgxPrivilege` are listed below.

- (0) `lgxPrivModify`
- (1) `lgxPrivDataFileCreateDelete`
- (2) `lgxPrivProgFileCreateDelete`
- (3) `lgxPrivLogicalWrite`
- (4) `lgxPrivPhysicalWrite`
- (5) `lgxPrivLogicalRead`
- (6) `lgxPrivPhysicalRead`
- (7) `lgxPrivModeChange`
- (8) `lgxPrivIOForce`
- (9) `lgxPrivSFCForce`
- (10) `lgxPrivClearMemory`
- (11) `lgxPrivDownload`
- (12) `lgxPrivOnlineEdit`
- (13) `lgxPrivEditPassword`

Class - The class from which the privilege status will be set.

Enabled - The status of the privilege. Enabled if `True`, Disabled if `False`.

Returns

Returns `True` if successful, otherwise returns `False`. If the currently active class does not have the privilege to modify privileges this function will return `False`.

Example

The following code snippet makes the call to `RSLogix` to disable the download privileges for class 3.

```
gPassPriv.SetFeaturePrivileges(lgxPrivDownload, 3, False)
```



SetProcessorPassword Boolean

Use this method to set the processor password initially or change the processor password if it is already set.

Syntax

`SetProcessorPassword(OldPassword as String, NewPassword as String)` as Boolean

Arguments

OldPassword - The string that contains the old password. If there is no old password, use an empty string.

NewPassword - The string that contains the new password. The password is limited to 10 characters in length.

Returns

True is returned if the new password is set, otherwise False is returned.

Example

The following code snippet makes the call to RSLogix to change the processor password from 'oldpasswd' to 'newpass.'

```
Result As Boolean
Result = gPassPriv.SetProcessorPassword("oldpasswd", "newpass")
```



SetProgFilePrivileges Boolean

Use this method to set the specified program file privileges in the current project.

Syntax

SetProgFilePrivileges(*ProgFile as Long, Class as Integer, Privileges as lgxPrivileges*) as Boolean

Arguments

ProgFile - The number of the program file that you wish to set class privileges for.

Class - The class number for which the privileges will be retrieved. The legal range is 1-4.

Privileges - The privileges that are enabled for the specified class and program file. Refer to valid lgxPrivilege types as listed at GetChannelPrivileges on page 177.

Returns

Returns True if successful, otherwise returns False. False will be returned if the class logged in to does not have "modify privilege" rights.

Example

The following code snippet makes the call to RSLogix to set the class 2 privileges for program file 7 to disable read and write privileges.

```
Rights As lgxPrivilege
Rights = lgxNoPriv
gPassPriv.SetProgFilePrivileges(7, 2, Rights)
```

Events

No events have been defined for the PasswordPrivilegeConfig object.

Appendix

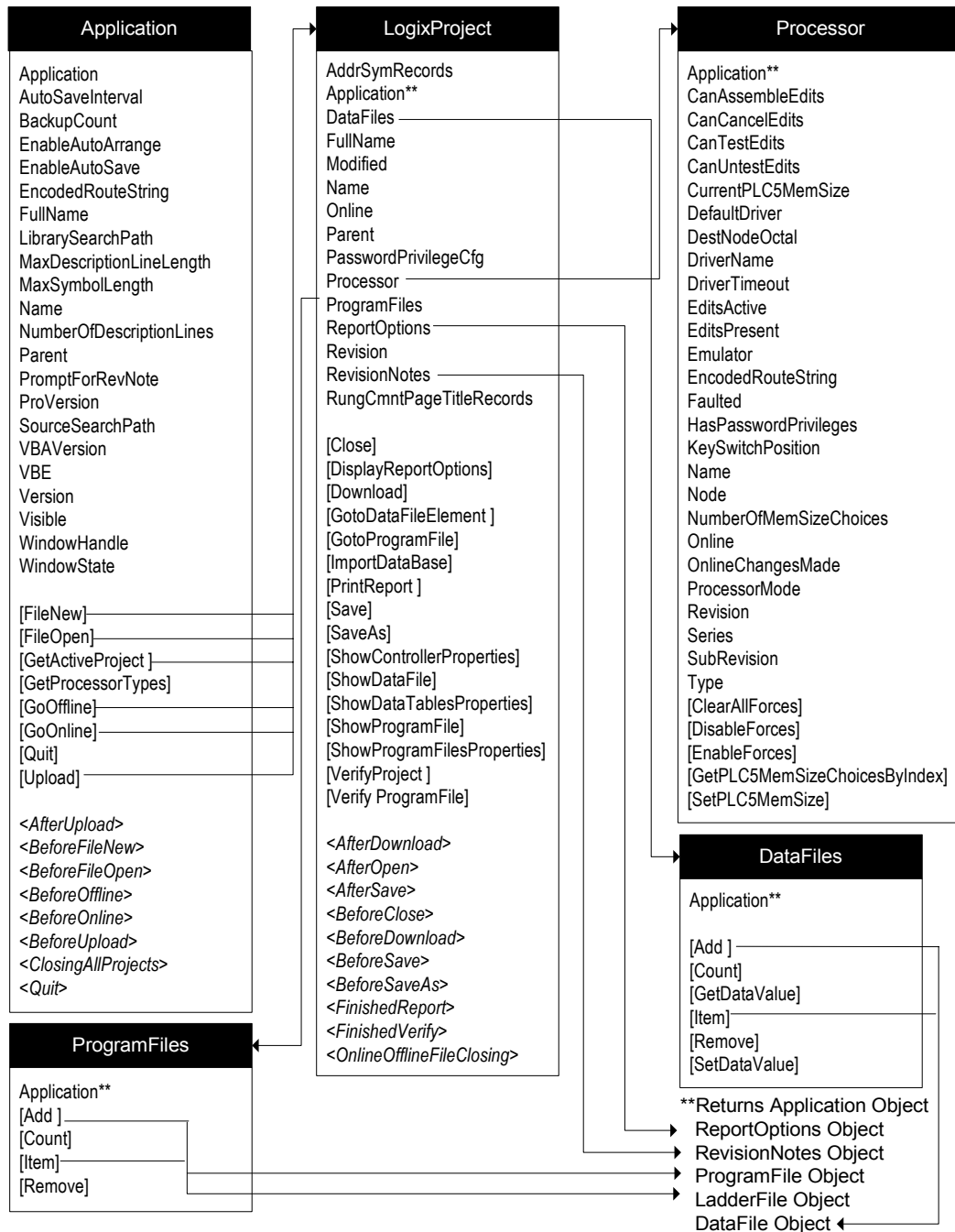
A

Object model diagrams

Introduction

The following pages illustrate the object models for RSLogix 5 and RSLogix 500 Programming Software.

RSLogix 5 object model summary



RSLogix 5 Object Model Summary, continued

DataFile
Application**
CanBeDeleted
CanBeMonitored
CanChangeScope
CanChangeSize
Debug
Description
FileName
FormattedName
GlobalScope
InUse
LocalScope
MaxDescriptionLength
MaxNameLength
Name
NumberOfElemets
Online
ReadPrivilege
Scopeable
Type
TypeAsString
WritePrivilege

ReportOptions
AddressSymbols
Application**
ChannelConfiguration
CrossReference
CrossReferenceByAddress
CrossReferenceFileEnd
CrossReferenceFileStart
CrossReferenceSymbolEnd
CrossReferenceSymbolStart
CustomDataMonitorFileRange
CustomDataMonitorFiles
DataFileList
DataFileRange
DataFiles
InstructionComments
IOInfo
MemoryUsage
MemoryUsageFileRange
ProcessorInfo
ProgramFileList
ProgramFileRange
ProgramFiles
SymbolGroups
TitlePage

ProgramFile
Application**
Debug
DefaultName
Description
FileNumber
FormattedName
InUse
MaxDescriptionLength
MaxNameLength
Name
Online
Programmable
ReadPrivilege
Type
WritePrivilege

PasswordPrivilegeConfig
Application**
CurrentClass
NodePrivilegeEntryCount
[AddNodePrivilegeEntry]
[ChangeNodePrivilegeInfo]
[ClassLogin]
[DownloadPrivChanges]
[GetChannelPrivileges]
[GetDataFilePrivileges]
[GetDefaultClass]
[GetFeaturePrivileges]
[GetNodePrivilegeInfo]
[GetProgFilePrivileges]
[IsClassPasswordProtected]
[RefreshChannelPrivsFromOnline]
[RefreshPassPrivsFromOnline]
[RemoveNodePrivilegeEntry]
[SetChannelPrivileges]
[SetClassPassword]
[SetDataFilePrivileges]
[SetDefaultClass]
[SetFeaturePrivileges]
[SetProcessorPassword]
[SetProgFilePrivileges]

RevisionNotes
Application**
InternalRevision
Revision
[Count]
[RevisionNote]

LadderFile
Application**
Debug
DefaultName
Description
EditsActive
FileNumber
Formattedname
InUse
MaxDescriptionLength
MaxNameLength
Name
Online
OnlineEdits
Programmable
RamEditsPending
ReadPrivilege
Reserved
Type
WritePrivilege
[GetRung]
[GetRungAsAscii]
[InsertRungAsAscii]
[NumberOfRungs]
[RemoveRung]

Rung
Active
Application**
Comment
DbaseID
EditsActive
EndRung
FileNumber
Modified
NumberOfInstructions
Online
Output
RungNumber
RungType
RungZoneDisplay
TempReplace
Title
Verified

KEY

Object

Property
[Method]
<Event>

RSLogix 5 Object Model Summary, Database Utilities

AddrSymRecord
Above
Address
Application**
Below
Description
DeviceCode
Scope
Symbol
SymbolGroup
[SetAbove]
[SetAddress]
[SetBelow]
[SetDescription]
[SetDeviceCode]
[SetScope]
[SetSymbol]
[SetSymGroup]

AddrSymRecords
Application**
Count
[Add]
[Duplicate]
[GetRecordIndexViaAddrOrSym]
[GetRecordViaAddrOrSym]
[GetRecordViaDesc]
[GetRecordViaIndex]
[RemoveRecordViaAddrOrSym]
[RemoveRecordViaIndex]
[SearchAndReplaceDesc]

RungCmntPageTitleRecord
Address
Application**
IsAttachedToAddress
PageTitle
ProgFile
RungComment
RungNumber
[SetAddress]
[SetPageTitle]
[SetProgFileAndRung]
[SetRungComment]

RungCmntPageTitleRecords
Application**
Count
[AddRecordAttachedToProgFileAndRung]
[AddRecordAttachedToAddress]
[DuplicateViaAddress]
[DuplicateViaFileRung]
[GetRecordViaAddress]
[GetRecordViaFileRung]
[GetRecordViaIndex]
[GetRecordViaPageTitle]
[GetRecordViaRungComment]
[RemoveRecordViaAddress]
[RemoveRecordViaFileRung]
[RemoveRecordViaIndex]
[SearchAndReplacePageTitle]
[SearchAndReplaceRungComment]

KEY

Object

Property

[Method]

<Event>

RSLogix 500 object model summary



RSLogix 500 Object Model Summary, continued

DataFile
Application**
CanBeDeleted
CanBeMonitored
CanChangeScope
CanChangeSize
Debug
Description
FileNumber
FormattedName
GlobalScope
InUse
LocalScope
MaxDescriptionLength
MaxNameLength
Name
NumberOfElements
Online
Reserved
Scopeable
Type
TypeAsString

ProgramFile
Application**
Debug
DefaultName
Description
FileNumber
FormattedName
InUse
MaxDescriptionLength
MaxNameLength
Name
Online
Programmable
ProtectionSupported
Reserved
Type

ReportOptions
AddressSymbols
Application**
ChannelConfiguration
CrossReference
CrossReferenceByAddress
CrossReferenceFileEnd
CrossReferenceFileStart
CrossReferenceSymbolEnd
CrossReferenceSymbolStart
CustomDataMonitorFileRange
CustomDataMonitorFiles
DataFileList
DataFileRange
DataFiles
InstructionComments
IOInfo
MemoryUsage
MemoryUsageFileRange
Multipoint
ProcessorInfo
ProgramFileList
ProgramFileRange
ProgramFiles
SymbolGroups
TitlePage

LadderFile
Application**
Debug
DefaultName
Description
EditsActive
FileNumber
FormattedName
InUse
MaxDescriptionLength
MaxNameLength
Name
Online
OnlineEdits
Programmable
ProtectionSupported
RamEditsPending
Reserved
Type
[GetRung]
[GetRungAsAscii]
[InsertRungAsAscii]
[NumberOfRungs]
[RemoveRung]

Rung
Active
Application**
Comment
DbaseID
EditsActive
EndRung
FileNumber
Modified
NumberOfInstructions
Online
Output
RungNumber
RungType
RungZoneDisplay
TempReplace
Title
Verified

RevisionNotes
Application**
InternalRevision
Revision
[Count]
[RevisionNote]

KEY
Object
Property
[Method]
<Event>

RSLogix 500 object model summary, database utilities

AddrSymRecord	AddrSymRecords	RungCmntPageTitleRecords
Above Address Application** Below Description DeviceCode Scope Symbol SymbolGroup [SetAbove] [SetAddress] [SetBelow] [SetDescription] [SetDeviceCode] [SetScope] [SetSymbol] [SetSymGroup]	Application** Count [Add] [Duplicate] [GetRecordIndexViaAddrOrSym] [GetRecordViaAddrOrSym] [GetRecordViaDesc] [GetRecordViaIndex] [RemoveRecordViaAddrOrSym] [RemoveRecordViaIndex] [SearchAndReplaceDesc]	Application** Count [AddRecordAttachedToProgFileAndRung] [AddRecordAttachedToAddress] [DuplicateViaAddress] [DuplicateViaFileRung] [GetRecordViaAddress] [GetRecordViaFileRung] [GetRecordViaIndex] [GetRecordViaPageTitle] [GetRecordViaRungComment] [RemoveRecordViaAddress] [RemoveRecordViaFileRung] [RemoveRecordViaIndex] [SearchAndReplacePageTitle] [SearchAndReplaceRungComment]
	RungCmntPageTitleRecord	
	Address Application** IsAttachedToAddress PageTitle ProgFile RungComment RungNumber [SetAddress] [SetPageTitle] [SetProgFileAndRung] [SetRungComment]	
		KEY
		Object
		Property
		[Method]
		<Event>

Appendix

B

Type definitions and constants

RSLogix 5 and RSLogix 500 type definitions and constants

When referring to the tables in this appendix, make sure to refer to the appropriate listing for either the RSLogix 5 or RSLogix 500 software product. Although the type definitions are similar their values differ.

IgxDataFileTypeConstants (RSLogix 5)

Used with the **DataFile** object and **DataFiles** collection. Not all may apply.

-1	IgxDTUNKNOWN	The type of data file is not recognized.
0	IgxDTBINARY	Indicates a Binary data file type.
1	IgxINTEGER	Indicates an Integer data file type.
2	IgxDTEXPANDER	Indicates an Expander data file type.
3	IgxDTASCII	Indicates the ASCII data file type.
4	IgxDTBCD	Indicates the Binary Coded Decimal data file type.
5	IgxDTSFCSTATUS	Indicates an SFC status data file type.
6	IgxDTSTRING	Indicates a String data file type.
7	IgxDTBLOCKXFER	Indicates a Block Transfer data file type.
8	IgxDTCONTROLNET	Indicates a ControlNet data file type.
9	IgxDTTIMER	Indicates a Timer data file type.
10	IgxDTCOUNTER	Indicates a Counter data file type.
11	IgxDTCONTROL	Indicates a Control data file type.
12	IgxDTTOKENDATA	Indicates a Token passing data file type.
13	IgxDTFLOAT	Indicates a Floating Point data file type.
14	IgxDTMESSAGE	Indicates a Message data file type.
15	IgxDTPIDBLOCK	Indicates a PID Block data file type.
16	IgxDTLONG	Indicates a Long data file type.
17	IgxDTMUTEX	A Mutex file type, handshaking between 2+> asynch threads.
18	IgxDTEVENT	Indicates an Event data file type.
19	IgxDTMANUALEVENT	Indicates a Manual Event data file type.
20	IgxDTDOUBLE	Indicates a Double data file type.
21	IgxDTTIME	A SoftLogix specific data type representing current date and time.
22	IgxDTINT64	Indicates a 64-bit Integer data file type.
23	IgxDTUNUSED	Indicates an unused data file.
24	IgxDTOUTPUT	Indicates an Output data file type.
25	IgxDTINPUT	Indicates an Input data file type.
26	IgxDTSTATUS	Indicates a status data file type.
27	IgxDTM0	An M0 file type. (Controls operation of devices on RIO link.)
28	IgxDTM1	An M1 file type (Status of devices on RIO link).
29	IgxDTSO	Indicates a SLC I/O data file.
30	IgxDTSI	Indicates a SLC I/O data file.
31	IgxLABEL	Indicates a Label file.
32	IgxDTSBR	Indicates a subroutine.
33	IgxDTRRET	Not useful to object model. Used internally only.
34	IgxDTBOOL	Indicates a Boolean data file type.
35	IgxDTNUMBEROFDATATYPES	Not useful to object model. Used internally only.
36	IgxDTRESERVED1	Reserved
37	IgxDTRESERVED2	Reserved
0xff	IgxDTSYSTEMTYPES	Not useful to object model. Used internally only.
0xffff	IgxDTMAXSYSTEMTYPE	Not useful to object model. Used internally only.
32768	IgxUSERTYPE0	Not useful to object model. Used internally only.

IgxDataFileTypeConstants (RSLogix 500)

Used with the **DataFile** object and **DataFiles** collection. Not all may apply.

-1	IgxDTUNKNOWN	The type of data file is not recognized.
0	IgxDTOUTPUT	Indicates an Output data file type.
1	IgxDTINPUT	Indicates an Input data file type.
2	IgxDTSTATUS	Indicates a status data file type.
3	IgxDTBINARY	Indicates a Binary data file type.
4	IgxDTTIMER	Indicates a Timer data file type.
5	IgxDTCOUNTER	Indicates a Counter data file type.
6	IgxDTCONTROL	Indicates a Control data file type.
7	IgxINTEGER	Indicates an Integer data file type.
8	IgxDTFLOAT	Indicates a Floating Point data file type.
9	IgxDTRESERVED1	Reserved
10	IgxDTUNUSED	Indicates an unused data file.
11	IgxDTRESERVED2	Reserved
12	IgxDTM1	An M1 file type (Status of devices on RIO link).
13	IgxDTM0	An M0 file type. (Controls operation of devices on RIO link.)
14	IgxDTSTRING	Indicates a String data file type.
15	IgxDTASCII	Indicates the ASCII data file type.
16	IgxDTLONG	Indicates a Long data file type.
17	IgxDTHSCOUNTER	Indicates a high speed counter data type.
18	IgxDTPULSE_TRAIN_OUT	Indicates a pulse train output data type.
19	IgxDTMESSAGE	Indicates a Message data file type.
20	IgxDTSEL_TIMED_INT	Indicates a selectable timed interrupt.
21	IgxDTEVENT_INPUT_INT	Indicates an event input interrupt.
22	IgxDTPIDBLOCK	Indicates a PID Block data file type.
23	IgxDTREAL_TIME_CLOCK	Indicates a real time clock data type.
24	IgxDTBASE_HARDWARE_INFO	Indicates the BHI (Base Hardware Information) function file.
25	IgxDTMEM_MODULE_INFO	Indicates the MMI (Memory Module Information) function file.
26	IgxDTDATA_ACCESS_TERM_INFO	Indicates the DAT (Data Access Terminal) function file.
27	IgxDTTRIM_POT_INFO	Indicates the TPI (Trim Pot Information) function file.
28	IgxDTCOM_STATUS	Indicates a communications status data file type.
29	IgxDTIOMOD_STATUS	Indicates an I/O module status data file type.
30	IgxDTPULSE_WIDTH_MOD	Indicates a pulse-width module data file type.
31	IgxDTDATA_LOG_STATUS	Indicates a data log status data file type.
32	IgxDTPLS	Indicates a Programmable Limit Switch data file type.
50	IgxDTBCD	Indicates the Binary Coded Decimal data file type.
51	IgxDTBLOCKXFER	Indicates a Block Transfer data file type.
52	IgxDTSFCSTATUS	Indicates an SFC status data file type.
53	IgxDTTOKENDATA	Indicates a Token passing data file type.
54	IgxDTCONTROLNET	Indicates a ControlNet data file type.
60	IgxLABEL	Indicates a Label file.
61	IgxDTSBR	Indicates a subroutine.
62	IgxDTRET	Not useful to object model. Used internally only.
64	IgxDTBOOL	Indicates a Boolean data file type.
32768	IgxDTUSERYPE0	Not useful to object model. Used internally only.

IgxKeyPositionConstants (RSLogix 5 and 500)

Used with the **Processor** object.

0	IgxUnknownKey	Processor keyswitch position is not known.
1	IgxKeyRemote	Processor keyswitch in Remote position.
2	IgxKeyProgram	Processor keyswitch in Program position.
3	IgxKeyRun	Processor keyswitch in Run position.

IgxOnlineAction (RSLogix 5 and 500)

Used with the **Application** and **LogixProject** objects.

1	IgxGoOnline	Instructs application to go online with the processor.
2	IgxGoOffline	Instructs application to go offline with the processor.

IgxProcessorTypeConstants (RSLogix 5)

Used with the **Application** and **Processor** objects.

-1	IgxLUNKNOWNPROC
1	IgxPLC_515
2	IgxPLC_512
3	IgxPLC_5VME
4	IgxPLC_525
5	IgxPLC_510
6	IgxPLC_540
7	IgxPLC_560
8	IgxPLC_540L
9	IgxPLC_560L
10	IgxPLC_530
11	IgxPLC_511
12	IgxPLC_520
13	IgxPLC_540VME
14	IgxPLC_540VMEL
15	IgxPLC_520E
16	IgxPLC_540E
17	IgxPLC_580
18	IgxPLC_516
19	IgxPLC_526
20	IgxPLC_536
21	IgxPLC_546
22	IgxPLC_546L
23	IgxPLC_566
24	IgxPLC_566L
25	IgxPLC_586
26	IgxPLC_580E
27	IgxPLC_530VME
28	IgxPLC_580VME
29	IgxPLC_520C
30	IgxPLC_540C
31	IgxPLC_560C
32	IgxPLC_580C
33	IgxPLC_520C2
34	IgxPLC_540C2
35	IgxPLC_560C2
36	IgxPLC_580C2
37	IgxPLC_526C2
38	IgxPLC_546C2
39	IgxSOFTLOGIX_5
40	IgxPLC_580VMEL

IgxProcessorTypeConstants (RSLogix 500)

Used with the `Application` and `Processor` objects.

-1	IgxLUNKNOWNPROC
0	Igx1747_L20A
1	Igx1747_L20B
2	Igx1747_L20C_F
3	Igx1747_L20D
4	Igx1747_L20E_G
5	Igx1747_L20L_N
6	Igx1747_L20P
7	Igx1747_L20R
8	Igx1747_L30A
9	Igx1747_L30B
10	Igx1747_L30C
11	Igx1747_L30D
12	Igx1747_L30L
13	Igx1747_L30P
14	Igx1747_L40A
15	Igx1747_L40B
16	Igx1747_L40C_F
17	Igx1747_L40E
18	Igx1747_L40L
19	Igx1747_L40P
20	Igx1747_L511
21	Igx1747_L514
26	Igx1747_L524
27	Igx1747_L532
28	IgxMICRO
29	Igx1747_L532B
30	Igx1747_L542A
31	Igx1747_L532C_D
32	Igx1747_L542B
33	Igx1747_L541
34	Igx1747_L543
36	IgxMICRO_DH485_
37	Igx1747_L551
38	Igx1747_L552
39	Igx1747_L553
40	Igx1747_L531
41	Igx1747_L532E
42	Igx1747_541C
43	Igx1747_542C
44	Igx1747_543C
45	IgxMICRO_ANALOG
46	Igx1747_551A
47	Igx1747_552A
48	Igx1747_553A
49	IgxMICRO1500LSP_A
50	IgxMICRO1200A
51	IgxMICRO1500LSP_B
52	IgxMICRO1500LRP_B
53	IgxMICRO1200B
54	IgxMICRO1500LSP_C
55	IgxMICRO1500LRP_C
57	IgxMICRO1200C
154	Igx1747_L531E
155	Igx1747_L551B
156	Igx1747_L552B
157	Igx1747_L553B

IgxProcOnlineState (RSLogix 5)

Used with the **Processor** and **LogixProject** objects. Not all apply. Refer to the chapter information for your specific use.

0	IgxOFFLINE	Processor mode is offline.
1	IgxDOWNLOAD	Processor mode is download.
2	IgxFAULTED	Processor mode is faulted.
3	IgxHARDPROGRAM	Processor mode is hard program.
4	IgxHARDTEST	Processor mode is hard test
5	IgxHARDRUN	Processor mode is hard run.
6	IgxREMOTEPROG	Processor mode is remote program.
7	IgxREMOTEST	Processor mode is remote test.
8	IgxREMOTERUN	Processor mode is remote run.
9	IgxBREAKPOINTSTOPPED	Processor mode is breakpoint stopped.

IgxProcOnlineState (RSLogix 500)

Used with the **Processor** and **LogixProject** objects. Not all apply. Refer to the chapter information for your specific use.

0	IgxOFFLINE	Processor mode is offline.
1	IgxDOWNLOAD	Processor mode is download.
2	IgxREMOTEPROG	Processor mode is remote program.
3	IgxSUSPEND	Suspend instruction executed while in remote program mode.
4	IgxREMOTERUN	Processor mode is remote run.
5	IgxTESTCONTINUOUS	Processor is in continuous test mode.
6	IgxTESTSINGLESCAN	Processor is in single scan test mode.
7	IgxTESTSTEPNOTRUNNING	Processor test step mode is not running.
8	IgxTESTSTEPRUNNING	Processor test step mode is running.
9	IgxHARDDOWNLOAD	Processor mode is hard download.
10	IgxHARDPROGRAM	Processor mode is hard program.
11	IgxHARDSUSPEND	Suspend instruction executed while in hard program mode.
12	IgxHARDRUN	Processor mode is hard run.
13	IgxFAULTED	Processor mode is faulted.

IgxProgramFileTypeConstants (RSLogix 5)

Used with the **ProgramFile** object and **ProgramFiles** collections. Not all apply. Refer to the chapter information for your specific use.

0	IgxHEADER	Indicates a header program file type.
1	IgxLADDER	Indicates a ladder program file type.
2	IgxSFCNEW	Indicates a new sequential function chart program file type.
3	IgxSFCOLD	Indicates an old sequential function chart program file type.
4	IgxSTX	Indicates a structured text program file type.
5	IgxIOFILE	Indicates an input/output program file type.
6	IgxSFCRTL	Not useful to object model. Used internally only.
7	IgxPLC_2	Indicates a PLC-2 program file type.
8	IgxCONFIG	Indicates a configuration file type.
9	IgxCAR	Indicates a CAR program file type.
10	IgxCOPROC	Not useful to object model. Used internally only.
11	IgxUNUSED	Indicates an unused program file.

IgxProgramFileTypeConstants (RSLogix 500)

Used with the **ProgramFile** object and **ProgramFiles** collections. Not all apply. Refer to the chapter information for your specific use..

0	IgxHEADER	Indicates a header program file type.
1	IgxLADDER	Indicates a ladder program file type.
2	IgxSFC	Indicates a sequential function chart program file type.
3	IgxUNUSED	Indicates an unused program file.

IgxRungZoneTypes (RSLogix 5 and 500)

Used with the **Rung** object.

0	IgxPlainRung	Indicates a plain, unedited rung.
1	IgxReplaceRung	Indicates a replaced rung.
2	IgxInsertRung	Indicates an inserted rung.
3	IgxDeleteRung	Indicates a deleted rung.
4	IgxEditRung	Indicates an edited rung.
5	IgxTmpInsertRung	Indicates a temporary inserted rung.
6	IgxTmpReplaceRung	Indicates a temporary replacement rung.
7	IgxAnyIRDRung	Indicates any type of rung.

IgxSaveAction (RSLogix 5 and 500)

Used with the **LogixProject** object. The only valid selections appear in the table below.

0	IgxNoAction	Indicates no external database files saved.
1	IgxSaveNativeExternalDB	Indicates database save to Native External file format.
2	IgxSaveAIExternalDB	Indicates database save to AI External file format.
3	IgxSaveAPSExternalDB	Indicates database save to APS External file format.

IgxUpDownAction (RSLogix 5 and 500)

Used with the **Application** object.

- | | | |
|---|--------------------|---|
| 1 | IgxUploadCreateNew | Upload and create new project. |
| 2 | IgxUploadCurrent | Upload and use the current project. |
| 3 | IgxUploadPath | Upload and use the project specified at path indicated. |

IgxWindowStateConstants (RSLogix 5 and 500)

Used with the **Application** object.

- | | | |
|---|-------------------------|--|
| 0 | IgxWindowStateNormal | Show application in normal window. |
| 1 | IgxWindowStateMinimized | Application is minimized to an icon. |
| 2 | IgxWindowStateMaximized | Application is maximized to full screen. |

IgxImportDBTypes (RSLogix 5 and 500)

Used with the **LogixProject** object.

- | | | |
|---|--------------------|-------------------------------------|
| 0 | IgxImportAddrSymDB | Import the address/symbol database. |
|---|--------------------|-------------------------------------|

IgxBinary (RSLogix 5)

Used with the **PasswordPrivilegeCfg** object.

- | | |
|---|-------------|
| 0 | IgxEnabled |
| 1 | IgxDisabled |
| 2 | IgxInvalid |

IgxChannel (RSLogix 5)

Used with the **PasswordPrivilegeCfg** object.

- | | | |
|---|-----------------|------------|
| 0 | IgxPLC5_Ch0 | Channel 0 |
| 1 | IgxPLC5_Ch1A | Channel 1A |
| 2 | IgxPLC5_Ch1B | Channel 1B |
| 3 | IgxPLC5_Ch2A | Channel 2A |
| 4 | IgxPLC5_Ch2B | Channel 2B |
| 5 | IgxPLC5_Ch2 | Channel 2 |
| 6 | IgxPLC5_Ch3A | Channel 3A |
| 7 | IgxPLC5_Offline | Offline |

IgxPrivilege (RSLogix 5)

Used with the **PasswordPrivilegeCfg** object.

- 0 IgxNoPriv
- 1 IgxReadPriv
- 2 IgxWritePriv
- 3 IgxReadWritePriv
- 4 IgxFailedToGetPriv

IgxPrivilegeType (RSLogix 5)

Used with the **PasswordPrivilegeCfg** object.

- 0 IgxPrivModify
- 1 IgxPrivDataFileCreateDelete
- 2 IgxPrivProgFileCreateDelete
- 3 IgxPrivLogicalWrite
- 4 IgxPrivPhysicalWrite
- 5 IgxPrivLogicalRead
- 6 IgxPrivPhysicalRead
- 7 IgxPrivModeChange
- 8 IgxPrivIOForce
- 9 IgxPrivSFCForce
- 10 IgxPrivClearMemory
- 11 IgxPrivDownload
- 12 IgxPrivOnlineEdit
- 13 IgxPrivEditPassword

IgxErrorTypes (RSLogix 5 and 500)

Refer to Appendix C for complete information about how you might use the IgxErrorType definition for error handling. .

Decimal Value	Error Type Definition	Hex Value
-2147220981	IgxError_Abort_Download	8004020B
-2147220957	IgxError_ACH_Load_Failed	80040223
-2147220964	IgxError_Autoconfig_Read_Failed	8004021C
-2147220923	IgxError_Automation_Inhibited	80040245
-2147220969	IgxError_Backup_Exists	80040217
-2147220934	IgxError_BOOTP_Cycle_Power	8004023A
-2147220991	IgxError_Cancel_Delete_All	80040201
-2147220966	IgxError_Cannot_Connect_To_Proc	8004021A
-2147220918	IgxError_Cannot_Find_Activation	8004024A
-2147220978	IgxError_Comm_Problem	8004020E
-2147220984	IgxError_Comms_Not_Set	80040208
-2147220932	IgxError_DataTable_Upload_Failed	8004023C
-2147220961	IgxDB_Create_Failed	8004021F
-2147220971	IgxError_DB_Error	80040215
-2147220942	IgxError_Default_Program	80040232
-2147220977	IgxError_Download_Failed	8004020F
-2147220975	IgxError_Enable_Forces_Failed	80040211
-2147220970	IgxError_Failed_Create_DB	80040216
-2147220973	IgxError_Failed_Del_Temp_DB	80040213
-2147220972	IgxError_Failed_Init_DB	80040214
-2147220968	IgxError_Failed_Online	80040218
-2147220924	IgxError_File_Is_Read_Only	80040244
-2147220976	IgxError_Forces_Exist	80040210
-2147220945	IgxError_General_IO_Error	8004022F
-2147220979	IgxError_Incompatible_Download_Types	8004020D
-2147220935	IgxError_Incorrect_Class_Priv	80040239
-2147220920	IgxError_Invalid_Argument	80040248
-2147220917	IgxError_Invalid_Class	8004024B
-2147220922	IgxError_Invalid_Data_File_Type	80040246
-2147220926	IgxError_Invalid_During_Compare	80040242
-2147220987	IgxError_Invalid_File_Extension	80040205
-2147220919	IgxError_Invalid_File_Size	80040249
-2147220967	IgxError_Invalid_IO_Object	80040219
-2147220925	IgxError_Invalid_Path_Specified	80040243
-2147220963	IgxError_Invalid_Rack	8004021D
-2147220962	IgxError_Invalid_Rack_Config	8004021E
-2147220958	IgxError_Invalid_RSS_File	80040222

continued on following page...

	...continued (IgxErrorTypes)	
-2147220921	IgxError_Invalid_Secure_Proc_Operation	80040247
-2147220955	IgxError_Invalid_SLC_File	80040225
-2147220952	IgxError_Library_Failure	80040228
-2147220956	IgxError_Library_Load_Failed	80040224
-2147220953	IgxError_Library_Partial_Load_Failed	80040227
-2147220954	IgxError_Library_Warnings_Exist	80040226
-2147220939	IgxError_Multiple_Files_Found	80040235
-2147220929	IgxError_Must_Be_Offline	8004023F
-2147220965	IgxError_No_AutoConfig	8004021B
-2147220928	IgxError_No_Checksum	80040240
-2147220982	IgxError_No_Controller_Response	8004020A
-2147220988	IgxError_No_Future_Access	80040204
-2147220940	IgxError_No_Match_Found	80040234
-2147220938	IgxError_No_Offline_DataFiles	80040236
-2147220947	IgxError_No_Processor	8004022D
-2147220990	IgxError_Not_Deletable	80040202
-2147220937	IgxError_Not_Done_Uploading	80040237
-2147220986	IgxError_Not_Offline	80040206
-2147220974	IgxError_Not_Program_Mode	80040212
-2147220989	IgxError_Not_Remote_Run	80040203
-2147220959	IgxError_Old_File_Format	80040221
-2147220927	IgxError_Online_Proc_Name_Invalid	80040241
-2147220933	IgxError_Open_Doc_Failed	8004023B
-2147220980	IgxError_Passwords_Dont_Match	8004020C
-2147220944	IgxError_Processor_Faulted	80040230
-2147220985	IgxError_Program_Errors	80040207
-2147220941	IgxError_Pswd_Failed_Login	80040233
-2147220950	IgxError_Rack_Out_Of_Range	8004022A
-2147220948	IgxError_Rack_Size_Out_Of_Range	8004022C
-2147220951	IgxError_RAM_Edits_Exist	80040229
-2147220943	IgxError_Remote_Emulator	80040231
-2147220930	IgxError_Save_In_Progress	8004023E
-2147220931	IgxError_Secure_Proc_Path_Not_Found	8004023D
-2147220960	IgxError_SLC_Func_Not_Available	80040220
-2147220949	IgxError_Slot_Out_Of_Range	8004022B
-2147220936	IgxError_STX_Not_Supported	80040238
-2147220992	IgxError_Unexpected	80040200
-2147220983	IgxError_Unknown_Proc	80040209
-2147220946	IgxError_Unsupported_Feature	8004022E

Handling errors

Versions 5.5 and greater of RSLogix have functionality (addition of the `lgxErrorType` type) that allows an automation client to determine which exceptions have been thrown.

The following example demonstrates how the `lgxErrorType` can be used to handle exceptions. Alternately you can choose to just check the error number using a `lgxErrorType` value.

```
Private Sub btnMakeVisible_Click()  
Dim ErrorType As RSLogix5.lgxErrorTypes  
ErrorType = lgxError_UNEXPECTED  
On Error GoTo Failed  
  
If g_Application.Visible = True Then  
    g_Application.Visible = False  
Else  
    g_Application.Visible = True  
End If  
Exit Sub  
  
Failed:  
MsgBox "Error # = " & Err.Number & " Error Desc = " & Err.Description  
If Err.Number = ErrorType Then  
    MsgBox "lgxError_UNEXPECTED"  
ElseIf Err.Number = lgxError_CANNOT_FIND_ACTIVATION Then  
    MsgBox "lgxError_CANNOT_FIND_ACTIVATION"  
End If  
  
End Sub
```

Any of the following lgxErrorTypes may be returned.

Decimal Value	Error Type Definition	Hex Value
-2147220981	IgxError_Abort_Download	8004020B
-2147220957	IgxError_ACH_Load_Failed	80040223
-2147220964	IgxError_Autoconfig_Read_Failed	8004021C
-2147220923	IgxError_Automation_Inhibited	80040245
-2147220969	IgxError_Backup_Exists	80040217
-2147220934	IgxError_BOOTP_Cycle_Power	8004023A
-2147220991	IgxError_Cancel_Delete_All	80040201
-2147220966	IgxError_Cannot_Connect_To_Proc	8004021A
-2147220918	IgxError_Cannot_Find_Activation	8004024A
-2147220978	IgxError_Comm_Problem	8004020E
-2147220984	IgxError_Comms_Not_Set	80040208
-2147220932	IgxError_DataTable_Upload_Failed	8004023C
-2147220961	IgxDB_Create_Failed	8004021F
-2147220971	IgxError_DB_Error	80040215
-2147220942	IgxError_Default_Program	80040232
-2147220977	IgxError_Download_Failed	8004020F
-2147220975	IgxError_Enable_Forces_Failed	80040211
-2147220970	IgxError_Failed_Create_DB	80040216
-2147220973	IgxError_Failed_Del_Temp_DB	80040213
-2147220972	IgxError_Failed_Init_DB	80040214
-2147220968	IgxError_Failed_Online	80040218
-2147220924	IgxError_File_Is_Read_Only	80040244
-2147220976	IgxError_Forces_Exist	80040210
-2147220945	IgxError_General_IO_Error	8004022F
-2147220979	IgxError_Incompatible_Download_Types	8004020D
-2147220935	IgxError_Incorrect_Class_Priv	80040239
-2147220920	IgxError_Invalid_Argument	80040248
-2147220917	IgxError_Invalid_Class	8004024B
-2147220922	IgxError_Invalid_Data_File_Type	80040246
-2147220926	IgxError_Invalid_During_Compare	80040242
-2147220987	IgxError_Invalid_File_Extension	80040205
-2147220919	IgxError_Invalid_File_Size	80040249
-2147220967	IgxError_Invalid_IO_Object	80040219
-2147220925	IgxError_Invalid_Path_Specified	80040243
-2147220963	IgxError_Invalid_Rack	8004021D
-2147220962	IgxError_Invalid_Rack_Config	8004021E
-2147220958	IgxError_Invalid_RSS_File	80040222
-2147220921	IgxError_Invalid_Secure_Proc_Operation	80040247
-2147220955	IgxError_Invalid_SLC_File	80040225
-2147220952	IgxError_Library_Failure	80040228
-2147220956	IgxError_Library_Load_Failed	80040224
-2147220953	IgxError_Library_Partial_Load_Failed	80040227
-2147220954	IgxError_Library_Warnings_Exist	80040226
-2147220939	IgxError_Multiple_Files_Found	80040235
-2147220929	IgxError_Must_Be_Offline	8004023F
-2147220965	IgxError_No_AutoConfig	8004021B
-2147220928	IgxError_No_Checksum	80040240
-2147220982	IgxError_No_Controller_Response	8004020A

continued on next page...

...continued

-2147220988	IgxError_No_Future_Access	80040204
-2147220940	IgxError_No_Match_Found	80040234
-2147220938	IgxError_No_Offline_DataFiles	80040236
-2147220947	IgxError_No_Processor	8004022D
-2147220990	IgxError_Not_Deletable	80040202
-2147220937	IgxError_Not_Done_Uploading	80040237
-2147220986	IgxError_Not_Offline	80040206
-2147220974	IgxError_Not_Program_Mode	80040212
-2147220989	IgxError_Not_Remote_Run	80040203
-2147220959	IgxError_Old_File_Format	80040221
-2147220927	IgxError_Online_Proc_Name_Invalid	80040241
-2147220933	IgxError_Open_Doc_Failed	8004023B
-2147220980	IgxError_Passwords_Dont_Match	8004020C
-2147220944	IgxError_Processor_Faulted	80040230
-2147220985	IgxError_Program_Errors	80040207
-2147220941	IgxError_Pswd_Failed_Login	80040233
-2147220950	IgxError_Rack_Out_Of_Range	8004022A
-2147220948	IgxError_Rack_Size_Out_Of_Range	8004022C
-2147220951	IgxError_RAM_Edits_Exist	80040229
-2147220943	IgxError_Remote_Emulator	80040231
-2147220930	IgxError_Save_In_Progress	8004023E
-2147220931	IgxError_Secure_Proc_Path_Not_Found	8004023D
-2147220960	IgxError_SLC_Func_Not_Available	80040220
-2147220949	IgxError_Slot_Out_Of_Range	8004022B
-2147220936	IgxError_STX_Not_Supported	80040238
-2147220992	IgxError_Unexpected	80040200
-2147220983	IgxError_Unknown_Proc	80040209
-2147220946	IgxError_Unsupported_Feature	8004022E

Appendix

D

General differences in the RSLogix 5 and 500 automation interfaces

There is significant commonality between the RSLogix 5 and RSLogix 500 object models. Exceptions are listed here.

PasswordPrivilegeConfig

All members of the PasswordPrivilegeConfig object apply to RSLogix 5 only.

DataFile object

Properties	RSLogix 5	RSLogix 500
NumberOfElements (integer)	Read/Write. Returns the number of elements in the data file.	Read Only.
Reserved (boolean)	Does not exist.	Read Only. Returns True if the data file is reserved.
ReadPrivilege (boolean)	Read Only. Returns whether the program file is read-enabled under the current privilege class.	Does not exist.
WritePrivilege (boolean)	Read Only. Returns whether the program file is write-enabled under the current privilege class.	Does not exist.

ProgramFile object

Properties	RSLogix 5	RSLogix 500
ProtectionSupported (boolean)	Does not exist.	Read only. Returns whether or not protection is supported by this program file.
Reserved (boolean)	Does not exist.	Read Only. Returns True if the program file is reserved.
ReadPrivilege (boolean)	Read Only. Returns whether the program file is read-enabled under the current privilege class.	Does not exist.
WritePrivilege (boolean)	Read Only. Returns whether the program file is write-enabled under the current privilege class.	Does not exist.

ReportOptions object

Properties	RSLogix 5	RSLogix 500
Multipoint (boolean)	Does not exist.	Read/Write. If set to True, a multipoint monitor report is included.

LogixProject object

Properties	RSLogix 5	RSLogix 500
PasswordPrivilegeCfg	Returns the Password/Privilege configuration for the RSLogix 5 processor.	Does not exist.

Processor object

Properties	RSLogix 5	RSLogix 500
CurrentPLC5MemSize (long)	Read only. Returns the processor memory size.	Does not exist.
HasPasswordPrivileges (boolean)	Read only. Returns if the processor supports privileges.	Does not exist.
NumberOfMemSizeChoices (short)	Read only. Returns how many choices of memory size you have for the current processor.	Does not exist.
ProgramID (integer)	Does not exist.	Read Only. Returns the 4-byte error check (CRC) of the program.
Series (integer)	Read/Write. Sets or returns the series # of the processor.	Does not exist.
Revision (integer)	Read/Write. Sets or returns the revision # of the processor.	Does not exist.
SubRevision (integer)	Read/Write. Sets or returns the subrevision # of the processor.	Does not exist.
Methods	RSLogix 5	RSLogix 500
GetPLC5MemSizeChoiceByIndex (long)	Gets the size of the processor's memory.	Does not exist.
SetPLC5MemSize	Sets the memory size of the processor.	Does not exist.

Ladder object

Properties	RSLogix 5	RSLogix 500
ProtectionSupported (Boolean)	Does not exist.	Read Only. Returns the attribute of protection supported by this ladder file.
ReadPrivilege (Boolean)	Read Only. Returns whether the ladder file is read-enabled under the current privilege class.	Does not exist.
WritePrivilege (Boolean)	Read Only. Returns whether the ladder file is write-enabled under the current privilege class.	Does not exist.

Index

A

Above property • 146
Active property • 110
Add method • 58, 76, 138
AddRecordAttachedToAddress method • 155
AddRecordAttachedToProgFileAndRung method • 154
Address property • 146, 166
AddressSymbols property • 128
AddrSymRecord
 about • 145
AddrSymRecord object • 145
 methods • 147
 properties • 146
AddrSymRecords
 about • 137
AddrSymRecords collection • 137
 methods • 138
 properties • 138
AddrSymRecords property • 26
AfterDownload event • 36
AfterOpen event • 36
AfterSave event • 37
AfterUpload event • 18
Application object • 9, 25, 45, 57, 65, 75, 85, 95, 109, 119, 127, 137, 145, 153, 165, 171
 about • 9
 events • 18
 example of use • 22
 methods • 13
 properties • 10
Application property • 10, 26, 46, 57, 66, 76, 86, 96, 110, 120, 128, 138, 146, 154, 166, 172
Automating the Documentation Database Editor • 4
Automating the Ladder Logic Editor • 3
AutoSaveInterval property • 11

B

BackupCount property • 11

BeforeClose event • 37
BeforeDownload event • 38
BeforeFileNew event • 19
BeforeFileOpen event • 19
BeforeOffline event • 20
BeforeOnline event • 20
BeforeSave event • 38
BeforeSaveAs event • 39
BeforeUpload event • 21
Below property • 146

C

CanAssembleEdits property • 46
CanBeDeleted property • 86
CanBeMonitored property • 86
CanCancelEdits property • 46
CanChangeScope property • 86
CanChangeSize property • 86
CanTestEdits property • 46
CanUntestEdits property • 46
ChannelConfiguration property • 128
Chapter summaries • 3
ClearAllForces method • 49
Close method • 28
ClosingAllProjects event • 21
Collections
 AddrSymRecords • 137
 DataFiles • 75
 ProgramFiles • 57
 RungCmntPageTitleRecords • 153
Comment property • 110
Constants (RSLogix 5) • 199
Constants (RSLogix 500) • 199
Count method • 59, 77, 120
Count property • 138, 154
CrossReference property • 128
CrossReferenceByAddress property • 129
CrossReferenceFileEnd property • 129
CrossReferenceFileStart property • 129
CrossReferenceSymbolEnd property • 129

CrossReferenceSymbolStart property • 129
CurrentClass property • 172
CurrentPLC5MemSize property • 47
CustomDataMonitorFileRange property • 130
CustomDataMonitorFiles property • 130

D

DataFile object • 85
 about • 85
 events • 88
 example of use • 89
 methods • 88
 properties • 86
DataFileList property • 130
DataFileRange property • 130
DataFiles collection • 75
 about • 75
 example of use • 79
 methods • 76
 properties • 75
DataFiles property • 27, 130
DbaseID property • 110
Debug property • 66, 86, 96
DefaultDriver property • 47
DefaultName property • 66, 96
Description property • 66, 86, 96, 146
DestNodeOctal property • 47
DeviceCode property • 146
Diagram of object models for 5 and 500 • 191
Differences between RSLogix 5 and 500 • 215
 DataFile object • 216
 Ladder object • 217
 LogixProject object • 216
 PasswordPrivilegeConfig object • 216
 Processor object • 217
 ProgramFile object • 216
 ReportOptions object • 216
DisableForces method • 50
DisplayReportOptions method • 28
Download method • 29
DriverName property • 47
DriverTimeout property • 47
Duplicate method • 139
DuplicateViaAddress method • 156
DuplicateViaFileRung method • 156

E

EditsActive property • 47, 96, 110
EditsPresent property • 47
Emulator property • 47
EnableAutoArrange property • 11
EnableAutoSave property • 11
EnableForces method • 50
EncodedRouteString property • 11, 47
EndRung property • 110
Error handling • 211
Examples
 Application object form and code • 22
 DataFile object form and code • 89
 DataFiles collection form and code • 79
 LadderFile object form and code • 101
 LogixProject object form and code • 41
 Processor object form and code • 51
 ProgramFile object form and code • 68
 ProgramFiles collection form and code • 60
 ReportOptions object form and code • 133
 RevisionNote object form and code • 121
 Rung object form and code • 112

F

Faulted property • 47
FileNew method • 13
FileNumber property • 66, 87, 96, 110
FileOpen method • 14
FinishedReport event • 39
FinishedVerify event • 40
FormattedName property • 67, 87, 97
FullName property • 11, 27

G

GetActiveProject method • 15
GetChannelPrivileges method • 176
GetDataValue method • 77
GetDefaultClass method • 178
GetFeaturePrivileges method • 179
GetPLC5MemSizeChoiceByIndex method • 50
GetProcessorTypes method • 15
GetRecordIndexViaAddrOrSym method • 139
GetRecordViaAddress method • 157
GetRecordViaAddrOrSym method • 140
GetRecordViaDesc method • 140
GetRecordViaFileRung method • 157

GetRecordViaIndex method • 141, 158
GetRecordViaPageTitle method • 159
GetRecordViaRungComment method • 159
GetRung method • 98
GetRungAsAscii method • 99
GlobalScope property • 87
GoOffline method • 15
GoOnline method • 16
GotoDataFileElement method • 29
GotoProgramFile method • 30
Graphical summary of object models • 191

H

HasPasswordPrivileges property • 48

I

Ideas about use • 2
ImportDataBase method • 31
InsertRungAsAscii method • 99
InstructionComments property • 130
InternalRevision property • 120
InUse property • 67, 87, 97
IOInfo property • 131
IsAttachedToAddress property • 166
IsClassPasswordProtected method • 182
Item method • 59, 77

K

KeySwitchPosition property • 48

L

LadderFile object • 95
 about • 95
 example of use • 101
 methods • 98
 properties • 96
lgxBinary
 (RSLogix 5) • 207
lgxChannel
 (RSLogix 5) • 207
lgxDataFileTypeConstants
 (RSLogix 5) • 200
 (RSLogix 500) • 201
lgxErrorTypes
 (RSLogix 5 and 500) • 209
lgxImportDBTypes

 (RSLogix 5 and 500) • 207
lgxKeyPositionConstants
 (RSLogix 5 and 500) • 202
lgxOnlineAction
 (RSLogix 5 and 500) • 202
lgxPrivilege
 (RSLogix 5) • 208
lgxPrivilegeType
 (RSLogix 5 and 500) • 208
lgxProcessorTypeConstants
 (RSLogix 5) • 203
 (RSLogix 500) • 204
lgxProcOnlineState
 (RSLogix 5) • 205
 (RSLogix 500) • 205
lgxProgramFileTypeConstant
 (RSLogix 5) • 206
 (RSLogix 500) • 206
lgxRungZoneTypes
 (RSLogix 5 and 500) • 206
lgxSaveAction
 (RSLogix 5 and 500) • 206
lgxUpDownloadAction
 (RSLogix 5 and 500) • 207
lgxWindowStateConstants
 (RSLogix 5 and 500) • 207
LibrarySearchPath property • 11
LocalScope property • 87
LogixProject object • 25
 about • 25
 events • 36
 example of use • 41
 methods • 28
 properties • 26

M

MaxDescriptionLength property • 67, 87, 97
MaxDescriptionLineLength property • 11
MaxNameLength property • 67, 87, 97
MaxSymbolLength property • 11
MemoryUsage property • 131
MemoryUsageFileRange property • 131
Modified property • 27, 110
Multipoint property • 131

N

Name property • 12, 27, 48, 67, 87, 97
Node property • 48
NodePrivilegeEntryCount property • 172
NumberOfDescriptionLines property • 12
NumberOfElements property • 87
NumberOfInstructions property • 110
NumberOfMemSizeChoices property • 48
NumberOfRungs method • 100

O

Object Model

graphically illustrated • 191

Objects

AddrSymRecord • 145
Application • 9, 25, 45, 57, 65, 75, 85, 95, 109, 119, 127, 137, 145, 153, 165, 171
DataFile • 85
LadderFile • 95
LogixProject • 25
PasswordPrivilegeConfig • 171
Processor • 45
ProgramFile • 65
ProgramFiles • 57
ReportOptions • 127
RevisionNotes • 119
Rung • 109
RungCmntPageTitleRecord • 165
OfflineClassLogin method • 175
Online property • 27, 48, 67, 87, 97, 111
OnlineChangesMade property • 48
OnlineEdits property • 97
OnlineOfflineFileClosing event • 40
Output property • 111

P

PageTitle property • 166
Parent property • 12, 27
PasswordPrivilegeCfg property • 27
PasswordPrivilegeConfig object • 171
about • 171
methods • 172
properties • 172
PrintReport method • 31
Processor object • 45
about • 45

example of use • 51
methods • 49
properties • 46

Processor property • 27
ProcessorInfo property • 131
ProcessorMode property • 48
ProgFile property • 166
ProgramFile object • 65
about • 65
example of use • 68
properties • 66
ProgramFileList property • 131
ProgramFileRange property • 132
ProgramFiles Collection • 57
ProgramFiles collection
about • 57
example of use • 60
methods • 58
properties • 57
ProgramFiles property • 27, 132
ProgramID property • 49
Programmable property • 67, 97
Programming tips • 6
PromptForRevNote property • 12
ProtectionSupported property • 67, 97
ProVersion property • 12

Q

Quit event • 21
Quit method • 17

R

RamEditsPending property • 97
ReadPrivilege property • 67, 88, 98
Remove method • 60, 78
RemoveRecordViaAddress method • 160
RemoveRecordViaAddrOrSym method • 142
RemoveRecordViaFileRung method • 161
RemoveRecordViaIndex method • 142, 161
RemoveRung method • 100
ReportOptions object • 127
about • 127
example of use • 133
properties • 128
ReportOptions property • 27
Reserved property • 67, 88, 98

- Revision property • 27, 49, 120
- RevisionNote method • 121
- RevisionNote object
 - example of use • 121
- RevisionNotes
 - methods • 120
- RevisionNotes object • 119
 - about • 119
 - properties • 120
- RevisionNotes property • 28
- Rung object • 109
 - about • 109
 - properties • 110
- RungCmntPageTitleRecord
 - about • 165
- RungCmntPageTitleRecord object • 165
 - methods • 167
 - properties • 166
- RungCmntPageTitleRecords collection • 153
 - about • 153
 - methods • 154
 - properties • 154
- RungCmntPageTitleRecords property • 28
- RungComment property • 166
- RungFile object
 - example of use • 112
- RungNumber property • 111, 166
- RungType property • 111
- RungZoneDisplay property • 111

S

- Save method • 32
- SaveAs method • 32
- Scope property • 147
- Scopeable property • 88
- SearchAndReplaceDesc method • 143
- SearchAndReplacePageTitle method • 162
- SearchAndReplaceRungComment method • 163
- Series property • 49
- SetAbove method • 147
- SetAddress method • 148, 167
- SetBelow method • 148
- SetClassPassword method • 185
- SetDataValue method • 78
- SetDefaultClass method • 187
- SetDescription method • 149
- SetDeviceCode method • 149
- SetFeaturePrivileges method • 187
- SetPageTitle method • 167
- SetPLC5MemSize method • 51
- SetProcessorPassword method • 188
- SetProgFileAndRung method • 168
- SetRungComment method • 168
- SetScope method • 151
- SetSymbol method • 151
- SetSymGroup method • 152
- ShowControllerProperties method • 33
- ShowDataFile method • 33
- ShowDataTablesProperties method • 34
- ShowProgramFile method • 34
- ShowProgramFilesProperties method • 34
- SourceSearchPath property • 12
- Subrevision property • 49
- Symbol property • 147
- SymbolGroup property • 147
- SymbolGroups property • 132

T

- TempReplace property • 112
- Tips • 6
- Title property • 112
- TitlePage property • 132
- Type definitions (RSLogix 5 and 500)
 - lgxErrorTypes • 209
 - lgxImportDBTypes • 207
 - lgxKeyPositionConstants • 202
 - lgxOnlineAction • 202
 - lgxRungZoneTypes • 206
 - lgxSaveAction • 206
 - lgxUpDownDownloadAction • 207
 - lgxWindowStateConstants • 207
- Type definitions (RSLogix 5) • 199
 - lgxBinary • 207
 - lgxChannel • 207
 - lgxDataFileTypeConstants • 200
 - lgxPrivilege • 208
 - lgxPrivilegeType • 208
 - lgxProcessorTypeConstants • 203
 - lgxProcOnlineState • 205
 - lgxProgramFileTypeConstants • 206
- Type definitions (RSLogix 500) • 199
 - lgxDataFileTypeConstants • 201

- lgxProcessorTypeConstants • 204
- lgxProcOnlineState • 205
- lgxProgramFileTypeConstants • 206
- Type property • 49, 68, 88, 98
- TypeAsString property • 88

U

- Upload method • 17
- Using this book • 3

V

- VBA
 - advantages • 1
 - illustration of environment • 2
- VBAVersion property • 12
- VBE property • 12
- Verified property • 112
- VerifyProgramFile method • 35
- VerifyProject method • 35
- Version property • 13
- Visible property • 13

W

- WindowHandle property • 13
- WindowsState property • 13
- WritePrivilege property • 68, 88, 98