RSLogix™
Automation Interface

Reference Manual

Doc ID LOGIX-RM001B-EN-P
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, RSVew32, RSVew, 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

Table of Contents
Chapter 4

**Processor object** ................................................................. 45
  Properties ........................................................................ 46
  Methods .......................................................................... 49
  Events ............................................................................ 51
  Summary example ............................................................ 51
  Form .............................................................................. 52
  Code ............................................................................... 53

Chapter 5

**ProgramFiles collection** ..................................................... 57
  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 collection** ........................................................... 75
  Properties ........................................................................ 75
  Methods .......................................................................... 76
  Events ............................................................................ 79
  Summary Example ........................................................... 79
  Form .............................................................................. 80
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
Chapter 12

**ReportOptions object** .................................................................127

- Properties ......................................................................................128
- Methods .........................................................................................132
- Events .............................................................................................132
- Summary example ........................................................................133

Form ..............................................................................................133
Code ..............................................................................................133

Chapter 13

**AddrSymRecords collection** .................................................137

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

Chapter 14

**AddrSymRecord object** .........................................................145

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

Chapter 15

**RungCmntPageTitleRecords collection** .........................153

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

Chapter 16

**RungCmntPageTitleRecord object** ........................................165

- Properties ......................................................................................166
- Methods .........................................................................................167
- Events .............................................................................................169
Chapter 17

**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
- lgxUpDownloadAction (RSLogix 5 and 500) ..............................................207
- lgxWindowStateConstants (RSLogix 5 and 500) .........................................207
- lgxBinary (RSLogix 5) ................................................................................207
- lgxChannel (RSLogix 5) ..............................................................................207
- lgxPrivilege (RSLogix 5) ...........................................................................208
Chapter 1

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.

<table>
<thead>
<tr>
<th>Objects</th>
<th>Purpose</th>
<th>Chpt:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>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.)</td>
<td>2</td>
</tr>
<tr>
<td>LogixProject</td>
<td>The LogixProject object represents the RSLogix project. Use it to access, define and return various attributes of an existing RSLogix project.</td>
<td>3</td>
</tr>
<tr>
<td>Processor</td>
<td>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.</td>
<td>4</td>
</tr>
<tr>
<td>ProgramFiles</td>
<td>The ProgramFiles collection represents all the program files in the project. Use it to add or remove program files from a collection.</td>
<td>5</td>
</tr>
<tr>
<td>ProgramFile</td>
<td>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.</td>
<td>6</td>
</tr>
<tr>
<td>Objects:</td>
<td>Purpose:</td>
<td>Chpt:</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>DataFiles</td>
<td>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.</td>
<td>7</td>
</tr>
<tr>
<td>DataFile</td>
<td>The DataFile object represents a data file in the project or processor. Use it to return defined attributes of the file.</td>
<td>8</td>
</tr>
<tr>
<td>LadderFile</td>
<td>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.</td>
<td>9</td>
</tr>
<tr>
<td>Rung</td>
<td>The Rung object represents a rung of ladder logic. Use it to obtain information about the rung.</td>
<td>10</td>
</tr>
<tr>
<td>RevisionNotes</td>
<td>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.</td>
<td>11</td>
</tr>
<tr>
<td>ReportOptions</td>
<td>The ReportOptions object represents the report settings associated with the project. Use it to read, establish or change settings.</td>
<td>12</td>
</tr>
<tr>
<td>PasswordPrivilegeConfig</td>
<td>The PasswordPrivilegeConfig object represents the master and class privilege administration unique to RSLogix5.</td>
<td>17</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Objects:</th>
<th>Purpose:</th>
<th>Chapter:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddrSymRecords</td>
<td>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.</td>
<td>13</td>
</tr>
<tr>
<td>AddrSymRecord</td>
<td>The AddrSymRecord object represents data in the address/symbol editor list. Use it to return or set a value in any field.</td>
<td>14</td>
</tr>
</tbody>
</table>
Supplemental information

The appendices in this book provide this additional information.

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

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:
   ```vba
   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.
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.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>FileNew</td>
<td>AfterUpload</td>
</tr>
<tr>
<td>AutoSaveInterval</td>
<td>FileOpen</td>
<td>BeforeFileNew</td>
</tr>
<tr>
<td>BackupCount</td>
<td>GetActiveProject</td>
<td>BeforeFileOpen</td>
</tr>
<tr>
<td>EnableAutoArrange</td>
<td>GetProcessorTypes</td>
<td>BeforeOffline</td>
</tr>
<tr>
<td>EnableAutoSave</td>
<td>GoOffline</td>
<td>BeforeOnline</td>
</tr>
<tr>
<td>EncodedRouteString</td>
<td>GoOnline</td>
<td>BeforeUpload</td>
</tr>
<tr>
<td>FullName</td>
<td>Quit</td>
<td>ClosingAllProjects</td>
</tr>
<tr>
<td>LibrarySearchPath</td>
<td>Upload</td>
<td>Quit</td>
</tr>
<tr>
<td>MaxDescriptionLineLength</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MaxSymbolLength</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NumberOfDescriptionLines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PromptForRevNote</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SourceSearchPath</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VBAVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VBE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visible</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WindowHandle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WindowState</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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).
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoSaveInterval</td>
<td>Long - Read/Write</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>BackupCount</td>
<td>Long - Read/Write</td>
<td></td>
<td>Returns the current backup count.</td>
</tr>
<tr>
<td>EnableAutoArrange</td>
<td>Boolean - Read/Write</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>EnableAutoSave</td>
<td>Boolean - Read/Write</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>EncodedRouteString</td>
<td>String - Read/Write</td>
<td></td>
<td>Internal use only.</td>
</tr>
<tr>
<td>FullName</td>
<td>String - Read Only</td>
<td></td>
<td>The full name of the application.</td>
</tr>
<tr>
<td>LibrarySearchPath</td>
<td>String - Read/Write</td>
<td></td>
<td>The path used for library files. This path should not exceed 256 characters.</td>
</tr>
<tr>
<td>MaxDescriptionLineLength</td>
<td>Long - Read/Write</td>
<td></td>
<td>Gets or sets the default length for descriptions used by the RSLogix database.</td>
</tr>
<tr>
<td>MaxSymbolLength</td>
<td>Long - Read/Write</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Access</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>------------</td>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read Only</td>
<td>The name of the application: “RSLogix 5” or “RSLogix 500.”</td>
</tr>
<tr>
<td>NumberOfDescriptionLines</td>
<td>Long</td>
<td>Read/Write</td>
<td>Gets or sets the default number of lines that RSLogix will accept for a description in its database.</td>
</tr>
<tr>
<td>Parent</td>
<td>Application</td>
<td>Read Only</td>
<td>Returns the parent of the Application object. This represents the entire RSLogix application.</td>
</tr>
<tr>
<td>PromptForRevNote</td>
<td>Boolean</td>
<td>Read/Write</td>
<td>Gets or sets the display of the revision note dialog when doing a “Save” or “Save As” operation in RSLogix.</td>
</tr>
<tr>
<td>ProVersion</td>
<td>Boolean</td>
<td>Read Only</td>
<td>Returns if this version of RSLogix is the “Pro” version or the “Standard” version.</td>
</tr>
<tr>
<td>SourceSearchPath</td>
<td>String</td>
<td>Read/Write</td>
<td>Gets or sets the path used for the searching of source projects. This path is used when going online, uploading, opening, and saving.</td>
</tr>
<tr>
<td>VBAVersion</td>
<td>String</td>
<td>Read Only</td>
<td>Returns the version of the VBA software currently running.</td>
</tr>
<tr>
<td>VBE</td>
<td>Object</td>
<td>Read Only</td>
<td>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.</td>
</tr>
</tbody>
</table>
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 lgxWindowStateConstants - Read/Write

Gets or sets the state of the main application window. These states are valid:

- (0) lgxWindowStateNormal - The display window is in its normal state.
- (1) lgxWindowStateMinimized - The display window has been minimized to an icon.
- (2) lgxWindowStateMaximized - 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 lgxProcessorTypeConstants, IgnorePrompts as Boolean, SaveChanges as Boolean) as LogixProject

Arguments

ProcessorType - When this argument is set to lgxUnknownProc 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
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.

```vba
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**

```vba
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.

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

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.

```vba
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, UploadAction as lgxUpDownloadAction, OnlineAction 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.

UploadAction - 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

OnlineAction - 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.

```vba
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.

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

**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

**BeforeFileOpen**

**Syntax**
BeforeFileOpen(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

**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.

```vbscript
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.

```vbscript
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.

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

Summary example

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.
Dim gApplication As RSLogix5.Application 'Application object
Dim gProject As RSLogix5.LogixProject 'LogixProject 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.
    gApplication.Visible = True
    gApplication.AutoSaveInterval = 3
    gApplicationWindowStateMaximized = 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.
End Sub
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
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.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddrSymRecords</td>
<td>Close</td>
<td>AfterDownload</td>
</tr>
<tr>
<td>Application</td>
<td>DisplayReportOptions</td>
<td>AfterOpen</td>
</tr>
<tr>
<td>DataFiles</td>
<td>Download</td>
<td>AfterSave</td>
</tr>
<tr>
<td>FullName</td>
<td>GotoDataFileElement</td>
<td>BeforeClose</td>
</tr>
<tr>
<td>Modified</td>
<td>GotoProgramFile</td>
<td>BeforeDownload</td>
</tr>
<tr>
<td>Name</td>
<td>ImportDataBase</td>
<td>BeforeSave</td>
</tr>
<tr>
<td>Online</td>
<td>PrintReport</td>
<td>FinishedReport</td>
</tr>
<tr>
<td>Parent</td>
<td>Save</td>
<td>OnlineOfflineFileClosing</td>
</tr>
<tr>
<td>PasswordPrivilegeCfg</td>
<td>SaveAs</td>
<td></td>
</tr>
<tr>
<td>Processor</td>
<td>ShowControllerProperties</td>
<td></td>
</tr>
<tr>
<td>ProgramFiles</td>
<td>ShowDataFile</td>
<td></td>
</tr>
<tr>
<td>ReportOptions</td>
<td>ShowDataTablesProperties</td>
<td></td>
</tr>
<tr>
<td>Revision</td>
<td>ShowProgramFile</td>
<td></td>
</tr>
<tr>
<td>RevisionNotes</td>
<td>ShowProgramFilesProperties</td>
<td></td>
</tr>
<tr>
<td>RungCmntPageTitleRecords</td>
<td>VerifyProject</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VerifyProgramFile</td>
<td></td>
</tr>
</tbody>
</table>
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.

```vba
' 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

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddrSymRecords</td>
<td>AddrSymRecord - Read Only Returns the AddrSymRecords collection.</td>
</tr>
<tr>
<td>Application</td>
<td>Application - Read Only Used without an object qualifier, this property returns an Application object that represents the RSLogix application.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>DataFiles</td>
<td>DataFiles - Read Only</td>
</tr>
<tr>
<td>FullName</td>
<td>String - Read Only</td>
</tr>
<tr>
<td>Modified</td>
<td>Boolean - Read Only</td>
</tr>
<tr>
<td>Name</td>
<td>String - Read Only</td>
</tr>
<tr>
<td>Online</td>
<td>Boolean - Read Only</td>
</tr>
<tr>
<td>Parent</td>
<td>Application - Read Only</td>
</tr>
<tr>
<td>PasswordPrivilegeCfg</td>
<td>PasswordPrivilegeCfg - Read Only</td>
</tr>
<tr>
<td>Processor</td>
<td>Processor - Read Only</td>
</tr>
<tr>
<td>ProgramFiles</td>
<td>ProgramFiles - Read Only</td>
</tr>
<tr>
<td>ReportOptions</td>
<td>ReportOptions - Read Only</td>
</tr>
<tr>
<td>Revision</td>
<td>Integer - Read Only</td>
</tr>
</tbody>
</table>
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
Use this method to download the current project into the processor.

Syntax
```
Download(IgnorePrompts as Boolean, OnlineAction as lgxOnlineAction,
ProcessorMode as lgxProcOnlineState) 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
Use this method to display the indicated data file element.

Syntax
```
GotoDataFileElement(Address as String) As Boolean
```

LogixProject object • 29
**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**

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)
```
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.

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

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.

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

**Save**

Use this method to save a project.

**Syntax**

```vbnet
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.

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

**SaveAs**

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

**Syntax**

```vbnet
SaveAs(IgnorePrompts as Boolean, AcceptDefaultActions as Boolean,
DBAction as lgxSaveAction, 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) lgxSaveAIExternalDB
  - (3) lgxSaveAPSExternalDB

```
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.

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

---

**ShowControllerProperties**

Use this method to display the controller properties dialog.

**Syntax**

```vba
ShowControllerProperties()
```

**Returns**
When successful this displays the controller properties dialog.

**Example**
The following code snippet displays the controller properties dialog.

```vba
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**

```vba
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.

```vba
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

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.ShowProgramFilesProperties()
**LogixProject object**

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

**Syntax**

```vba
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.

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

---

**VerifyProgramFile**

Use this method to verify a designated program file.

**Syntax**

```vba
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.

```vba
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

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

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

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

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
gProjEvents_BeforeSave = False
End Function

**BeforeSaveAs**

**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.

```vbscript
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.

```vbscript
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.

```vba
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.

```vba
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.

![RSLogix VB Example Form](image)

**Code**

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

Dim gApplication As RSLogix5.Application 'Application object
Dim gProject As RSLogix5.LogixProject 'LogixProject object`
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
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.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>ClearAllForces</td>
</tr>
<tr>
<td>CanAssembleEdits</td>
<td>DisableForces</td>
</tr>
<tr>
<td>CanCancelEdits</td>
<td>EnableForces</td>
</tr>
<tr>
<td>CanTestEdits</td>
<td>GetPLC5MemSizeChoiceByIndex - (RSLogix 5 only)</td>
</tr>
<tr>
<td>CanUntestEdits</td>
<td>SetPLC5MemSize - (RSLogix 5 only)</td>
</tr>
<tr>
<td>DefaultDriver</td>
<td></td>
</tr>
<tr>
<td>CurrentPLC5MemSize - (RSLogix 5 only)</td>
<td></td>
</tr>
<tr>
<td>DestNodeOctal</td>
<td></td>
</tr>
<tr>
<td>DriverName</td>
<td></td>
</tr>
<tr>
<td>DriverTimeout</td>
<td></td>
</tr>
<tr>
<td>EditsActive</td>
<td></td>
</tr>
<tr>
<td>EditsPresent</td>
<td></td>
</tr>
<tr>
<td>Emulator</td>
<td></td>
</tr>
<tr>
<td>EncodedRouteString</td>
<td></td>
</tr>
<tr>
<td>Faulted</td>
<td></td>
</tr>
<tr>
<td>HasPasswordPrivileges - (RSLogix 5 only)</td>
<td></td>
</tr>
<tr>
<td>KeySwitchPosition</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Node</td>
<td></td>
</tr>
<tr>
<td>NumberOfMemSizeChoices - (RSLogix 5 only)</td>
<td></td>
</tr>
<tr>
<td>Online</td>
<td></td>
</tr>
<tr>
<td>OnlineChangesMade</td>
<td></td>
</tr>
<tr>
<td>ProcessorMode</td>
<td></td>
</tr>
<tr>
<td>ProgramID - (RSLogix 500 only)</td>
<td></td>
</tr>
<tr>
<td>Revision - (RSLogix 5 only)</td>
<td></td>
</tr>
<tr>
<td>Series - (RSLogix 5 only)</td>
<td></td>
</tr>
<tr>
<td>SubRevision - (RSLogix 5 only)</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Events</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-None-</td>
<td></td>
</tr>
</tbody>
</table>
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.

```vba
'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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Read/Write</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application</strong></td>
<td>Application - Read Only</td>
<td></td>
</tr>
<tr>
<td>This property returns an Application object that represents the RSLogix 5 or RSLogix 500 application.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CanAssembleEdits</strong></td>
<td>Boolean - Read Only</td>
<td></td>
</tr>
<tr>
<td>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).</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CanCancelEdits</strong></td>
<td>Boolean - Read Only</td>
<td></td>
</tr>
<tr>
<td>Queries and returns whether or not edits can be cancelled in the processor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CanTestEdits</strong></td>
<td>Boolean - Read Only</td>
<td></td>
</tr>
<tr>
<td>Queries and returns whether or not edits can be tested in the processor (examine how the program operates with the edited rung).</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CanUntestEdits</strong></td>
<td>Boolean - Read Only</td>
<td></td>
</tr>
<tr>
<td>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).</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Processor object

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentPLC5MemSize</td>
<td>Long</td>
<td>Read/Write</td>
<td>This property returns the value of the current processor memory size in bytes.</td>
</tr>
<tr>
<td>DefaultDriver</td>
<td>Boolean</td>
<td>Read Only</td>
<td>Queries and returns whether or not the default driver is being used in the current project.</td>
</tr>
<tr>
<td>DestNodeOctal</td>
<td>Boolean</td>
<td>Read Only</td>
<td>Returns whether or not the destination node is “expressed in” or “expected to be in” octal.</td>
</tr>
<tr>
<td>DriverName</td>
<td>String</td>
<td>Read Only</td>
<td>Returns the name of the communications driver currently being used to communicate with the processor.</td>
</tr>
<tr>
<td>DriverTimeout</td>
<td>Integer</td>
<td>Read/Write</td>
<td>Returns or sets the timeout expressed in seconds for the communication driver.</td>
</tr>
<tr>
<td>EditsActive</td>
<td>Boolean</td>
<td>Read Only</td>
<td>Queries and returns whether or not edits are active in the processor.</td>
</tr>
<tr>
<td>EditsPresent</td>
<td>Boolean</td>
<td>Read Only</td>
<td>Queries and returns whether or not edits are present in the processor.</td>
</tr>
<tr>
<td>Emulator</td>
<td>Boolean</td>
<td>Read Only</td>
<td>Queries and returns whether or not the emulator is being used instead of a real processor.</td>
</tr>
<tr>
<td>EncodedRouteString</td>
<td>String</td>
<td>Read/Write</td>
<td>This property is for internal use only.</td>
</tr>
<tr>
<td>Faulted</td>
<td>Boolean</td>
<td>Read Only</td>
<td>Queries and returns whether or not the processor is faulted.</td>
</tr>
</tbody>
</table>

(5 only)
HasPasswordPrivileges Boolean - Read Only

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

KeySwitchPosition lgxKeyPositionConstants - 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) lgxUnknownKey
- (1) lgxKeyRemote
- (2) lgxKeyProgram
- (3) lgxKeyRun

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.

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 lgxProcOnlineState - 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) lgxRemoteProg
- (7) lgxRemoteTest
- (8) lgxRemoteRun
See Appendix B for a complete list of type definitions for lgxProcOnlineState.

(500 only)

**ProgramID**

Integer - Read Only

Returns the 4-byte error check (CRC) of the program.

(5 only)

**Revision**

Integer - Read/Write

Sets or returns the revision number of the processor.

(5 only)

**Series**

Integer - Read/Write

Sets or returns the series of the processor.

(5 only)

**Subrevision**

Integer - Read/Write

Sets or returns the subrevision of the processor.

**Type**

lgxProcessorTypeConstant - Read Only

Returns the type of the processor as a lgxProcessorTypeConstant.

### 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()

GetPLC5MemSizeChoiceByIndex

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)

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.

```vbnet
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**

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.
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.
    gApplication.Visible = True
    gApplication.AutoSaveInterval = 3
    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
    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).
End Sub
' 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
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

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.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Add</td>
<td>-None-</td>
</tr>
<tr>
<td></td>
<td>Count</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Item</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Remove</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Application - Read Only</th>
</tr>
</thead>
</table>

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

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

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

**Syntax**

```vba
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.

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

### Item

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

**Syntax**

```vba
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.

```vba
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

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.

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
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 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

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

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()
DimReturnValue 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
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.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>-None-</td>
<td>-None-</td>
</tr>
<tr>
<td>Debug</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DefaultName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FormattedName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>InUse</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MaxDescriptionLength</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MaxNameLength</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Online</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProtectionSupported</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReadPrivilege</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WritePrivilege</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The following commented code example illustrates a typical call to the main program file (ladder file #2).

```vba
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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application</strong></td>
<td>Application</td>
<td>Read Only</td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td><strong>DefaultName</strong></td>
<td>String</td>
<td>Read Only</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>String</td>
<td>Read/Write</td>
</tr>
<tr>
<td><strong>FileNumber</strong></td>
<td>Long</td>
<td>Read Only</td>
</tr>
</tbody>
</table>

- **Application**: This property returns the Application object.
- **Debug**: This property returns whether or not the program file is for debug use only.
- **DefaultName**: This property returns the default name of the file.
- **Description**: This property returns or sets the description of the program file.
- **FileNumber**: 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.

(500 only) ProtectionSupported Boolean - Read Only

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

(5 only) 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.

(500 only) Reserved Boolean - Read Only

This property returns True if the program file is reserved.
This property returns the type of file. Possible returned types are listed below and described in Appendix B.

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

(5 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

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.
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
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

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

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
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

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Add</td>
<td>-None-</td>
</tr>
<tr>
<td></td>
<td>Count</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GetDataValue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Item</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Remove</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SetDataValue</td>
<td></td>
</tr>
</tbody>
</table>

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

```vba
'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.
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**

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

**Syntax**

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

**Arguments**

- **FileNumber** - 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**

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.

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

---

**GetDataValue**

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.

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

---

**Item**

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.

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

Remove
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.

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

SetDataValue
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

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.
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

errorHandler:

End Sub
\begin{verbatim}
' (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

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

Private Sub Text1_Click()
\end{verbatim}
' 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
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.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>-None-</td>
<td>-None-</td>
</tr>
<tr>
<td>CanBeDeleted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CanBeMonitored</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CanChangeScope</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CanChangeSize</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debug</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FormattedName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GlobalScope</td>
<td></td>
<td></td>
</tr>
<tr>
<td>InUse</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LocalScope</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MaxDescriptionLength</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MaxNameLength</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NumberOfElements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Online</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReadPrivilege</td>
<td>(RSLogix 5 only)</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>(RSLogix 500 only)</td>
<td></td>
</tr>
<tr>
<td>Scopeable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TypeAsString</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WritePrivilege</td>
<td>(RSLogix 5 only)</td>
<td></td>
</tr>
</tbody>
</table>
The following commented code example illustrates how you might access the DataFile object.

```vba
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

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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Read/Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Application</td>
<td>Read Only</td>
</tr>
<tr>
<td>CanBeDeleted</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td>CanBeMonitored</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td>CanChangeScope</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td>CanChangeSize</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td>Debug</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>Read/Write</td>
</tr>
</tbody>
</table>

- **Application**
  - Returns an Application object that represents the RSLogix application.
- **CanBeDeleted**
  - Returns whether or not the data file may be deleted.
- **CanBeMonitored**
  - Returns whether or not the data file may be monitored.
- **CanChangeScope**
  - Returns whether or not the scope of this file can be changed.
- **CanChangeSize**
  - Returns whether or not the file can have the number of elements changed.
- **Debug**
  - Returns whether or not the file is for debug use only.
- **Description**
  - Represents the text description of this data file.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileNumber</td>
<td>Integer - Read Only</td>
<td></td>
<td>Returns the file number of this data file.</td>
</tr>
<tr>
<td>FormattedName</td>
<td>String - Read Only</td>
<td></td>
<td>Returns the full formatted name of the data file.</td>
</tr>
<tr>
<td>GlobalScope</td>
<td>Boolean - Read Only</td>
<td></td>
<td>Returns whether or not this data file is of global scope.</td>
</tr>
<tr>
<td>InUse</td>
<td>Boolean - Read Only</td>
<td></td>
<td>Returns whether or not this file is being used.</td>
</tr>
<tr>
<td>LocalScope</td>
<td>Boolean - Read Only</td>
<td></td>
<td>Returns whether or not this data file is of local scope.</td>
</tr>
<tr>
<td>MaxDescriptionLength</td>
<td>Integer - Read Only</td>
<td></td>
<td>Returns the maximum number of characters for the file description.</td>
</tr>
<tr>
<td>MaxNameLength</td>
<td>Integer - Read Only</td>
<td></td>
<td>Returns the maximum number of characters for the file name.</td>
</tr>
<tr>
<td>Name</td>
<td>String - Read/Write</td>
<td></td>
<td>Returns or sets the name of the file</td>
</tr>
</tbody>
</table>
| 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.            |
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.

This property returns True if the data file is reserved.

Returns whether or not this file can be scoped.

Returns the type of data file as a lgxDataFileTypeConstants. The valid selections are listed and defined in Appendix B.

Returns the type of data file as a text string.

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.

There are no methods defined for the DataFile object.

There are no events defined for the DataFile object.
Summary Example

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.

![Form Image]

**Code**

```vba
'-------------------------------
' 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
```
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.WindowsState = 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()
DimReturnValue 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
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
- Application
- Debug
- DefaultName
- Description
- EditsActive
- FileNumber
- FormattedName
- InUse
- MaxDescriptionLength
- MaxNameLength
- Name
- Online
- OnlineEdits
- Programmable
- ProtectionSupported (500 only)
- RamEditsPending
- ReadPrivilege (5 only)
- Reserved
- Type
- WritePrivilege (5 only)

Methods
- GetRung
- GetRungAsAscii
- InsertRungAsAscii
- NumerOfRungs
- RemoveRung

Events
- None-
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.

```vbscript
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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Read/Write</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application</strong></td>
<td>Application - Read Only</td>
<td></td>
</tr>
<tr>
<td>This property returns the application object.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td>Boolean - Read Only</td>
<td></td>
</tr>
<tr>
<td>This property returns whether or not the ladder file is for debugging use only.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DefaultName</strong></td>
<td>String - Read Only</td>
<td></td>
</tr>
<tr>
<td>This property returns the default name of the ladder file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>String - Read/Write</td>
<td></td>
</tr>
<tr>
<td>This property returns or sets the description string for the ladder file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>EditsActive</strong></td>
<td>Boolean - Read Only</td>
<td></td>
</tr>
<tr>
<td>This property returns whether or not there are any edits active in the processor that have not been assembled</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FileNumber</strong></td>
<td>Long - Read Only</td>
<td></td>
</tr>
<tr>
<td>This property returns the number of the ladder file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Access</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------</td>
<td>----------------</td>
</tr>
<tr>
<td>FormattedName</td>
<td>String</td>
<td>Read Only</td>
</tr>
<tr>
<td>InUse</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td>MaxDescriptionLength</td>
<td>Long</td>
<td>Read Only</td>
</tr>
<tr>
<td>MaxNameLength</td>
<td>Long</td>
<td>Read Only</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Online</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td>OnlineEdits</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td>Programmable</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td>ProtectionSupported</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td>RamEditsPending</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
</tbody>
</table>
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  lgxProgramFileTypeConstants - Read Only

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

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(RunNumber as Long) as Rung

Arguments

RunNumber - 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.

```vba
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.

```vba
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")
```

![Diagram of a rung inserted into ladder logic]

**NumberOfRungs**
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**
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

<table>
<thead>
<tr>
<th>Important</th>
<th>This book assumes that you have the basic knowledge required to work with forms and controls in Visual Basic.</th>
</tr>
</thead>
</table>

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 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

End Sub

errorHandler:

' 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.
ReturnVal =gLadderFile.InsertRungAsAscii(CInt(Text8.Text), Text7.Text)
MsgBox "Returned: " & ReturnVal
End Sub
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.

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

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

The Properties, Methods, and Events are as follows:

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>-None-</td>
<td>-None-</td>
</tr>
<tr>
<td>Application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DbaseID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EditsActive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EndRung</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modified</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NumberOfInstructions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Online</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RungNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RungType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RungZoneDisplay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TempReplace</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Title</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verified</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
**lgxRungZoneTypes - 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) lgxPlainRung
- (1) lgxReplaceRung
- (2) lgxInsertRung
- (3) lgxDeleteRung
- (4) lgxEditRung

### RungZoneDisplay
**lgxRungZoneTypes - 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) lgxPlainRung
- (1) lgxReplaceRung
- (2) lgxInsertRung
- (3) lgxDeleteRung
- (4) lgxEditRung
- (5) lgxTmpInsertRung
- (6) lgxTmpReplaceRung
- (7) lgxAnyIrdRung
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.
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
DimgLadderFile 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
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

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

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()
DimReturnValueAsBoolean

'Addarungdefinedbyuserenteredtextanddisplaythereturn
'valueinamessagebox.
ReturnValue = gLadderFile.InsertRungAsAscii(CInt(Text8.Text),
Text7.Text)
MsgBox"Returned: " & ReturnValue
EndSub

'-----------------------------------------------
'RungObject
'-----------------------------------------------

PrivateSubCommand13_Click()
'Sets the global rung object to therung number specified by atextbox.
SetgRung=gLadderFile.GetRung(CInt(Text11.Text))

'Assigntherungtitleandcommentpropertytotheirrespective
'textboxes.
Text9.Text = gRung.Title
Text10.Text = gRung.Comment

EndSub
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.

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.

```vbnet
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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Application</td>
<td>Read Only</td>
</tr>
<tr>
<td>InternalRevision</td>
<td>Integer</td>
<td>Read Only</td>
</tr>
<tr>
<td>Revision</td>
<td>Long</td>
<td>Read Only</td>
</tr>
</tbody>
</table>

- **Application**
  This property returns the Application object.

- **InternalRevision**
  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**
  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**
  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.

  ```vbnet
  Application Application - Read Only
  InternalRevision Integer - Read Only
  Revision Long - Read Only
  Count Integer
  ```
MsgBox "There are " & gRevisionNotes.Count & " revision notes."

<table>
<thead>
<tr>
<th>RevisionNote</th>
<th>String</th>
</tr>
</thead>
</table>

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.

```vbnet
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.

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
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

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

' LogixProject
' RevisionNotes
errhand:
    MsgBox "That revision note was not found."
End Sub
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.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddressSymbols</td>
<td>-None-</td>
<td>-None-</td>
</tr>
<tr>
<td>Application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ChannelConfiguration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CrossReference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CrossReferenceByAddress</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CrossReferenceFileEnd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CrossReferenceFileStart</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CrossReferenceSymbolEnd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CrossReferenceSymbolStart</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CustomDataMonitorFileRange</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CustomDataMonitorFiles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DataFileList</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DataFileRange</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DataFiles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>InstructionComments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOLinfo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MemoryUsage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MemoryUsageFileRange</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multipoint (RSLogix 500 only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProcessorInfo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProgramFileList</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProgramFileRange</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProgramFiles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SymbolGroups</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TitlePage</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The following commented code example illustrates how to access the Report Options object.

```vba
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.
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”

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.

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

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”

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.

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.
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.

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.

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”

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

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.

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

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

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.

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.

There are no methods defined for the ReportOptions object.

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.

![Form screenshot]

Code
'-----------------------------------
' Global variables
'-----------------------------------

Dim gApplication As RSLogix5.Application 'Application object
Dim gProject As RSLogix5.LogixProject 'LogixProject object
Dim gReportOptions As RSLogix5.ReportOptions 'ReportOptions Object
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:
On Error Resume Next

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()
    DimReturnValue 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
    DimReturnValue 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
Private Sub Command7_Click()
    DimReturnValue 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
Chapter 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

- Application Count

Methods

- Add
- Duplicate
- GetRecordIndexViaAddrOrSym
- GetRecordViaAddrOrSym
- GetRecordViaDesc
- GetRecordViaIndex
- RemoveRecordViaAddrOrSym
- RemoveRecordViaIndex
- SearchAndReplaceDesc

Events

- None

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

```vbnet
' 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
```

AddrSymRecords collection • 137
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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application</strong></td>
<td>Application - Read Only</td>
</tr>
<tr>
<td>This property returns an Application that represents the RSLogix application.</td>
<td></td>
</tr>
<tr>
<td><strong>Count</strong></td>
<td>Long - Read Only</td>
</tr>
<tr>
<td>This property returns the number of records in the Address/Symbol database.</td>
<td></td>
</tr>
</tbody>
</table>

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**

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.

```vba
Dim Index As Long
Index = gAddrSymRecords.GetRecordIndexViaAddrOrSym("B3:0",0)
```

**GetRecordViaAddrOrSym**  

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.

```vba
gAddrSymRecord = gAddrSymRecords.GetRecordViaAddrOrSym("B3:0",0)
```

**GetRecordViaDesc**  

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.

```vbnet
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.

```vba
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.
Chapter 14

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.

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

```vba
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
End Sub
```

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

This property returns a string that identifies the symbol of the AddrSymRecord.

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**

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.

```vbnet
Result As Boolean
Result = gAddrSymRecord.SetAbove("abovetext")
```

**SetAddress**

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.

```vbnet
Result As Boolean
Result = gAddrSymRecord.SetAddress("B3:0")
```

**SetBelow**

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.”

```vba
Result As Boolean
Result = gAddrSymRecord.SetBelow("TEST")
```

**SetDescription**  
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.”

```vba
Result As Boolean
Result = gAddrSymRecord.SetDescription("Test Description")
```

**SetDeviceCode**  
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

<table>
<thead>
<tr>
<th>Input Device Codes</th>
<th>2KNCL</th>
<th>3SSNCL</th>
<th>MMPNCA</th>
<th>PBNCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>2KNCM</td>
<td>3SSNCL</td>
<td>MMPNCG</td>
<td>PBNCR</td>
<td></td>
</tr>
<tr>
<td>2KNOR</td>
<td>3SSNOR</td>
<td>MMPNOR</td>
<td>PBNOG</td>
<td></td>
</tr>
<tr>
<td>2SSNCL</td>
<td>CRNC</td>
<td>MOPNCA</td>
<td>PBNOE</td>
<td></td>
</tr>
<tr>
<td>2SSNCR</td>
<td>FSNC</td>
<td>MOPNCR</td>
<td>PBNOQ</td>
<td></td>
</tr>
<tr>
<td>2SNOL</td>
<td>FSNO</td>
<td>MOPNOA</td>
<td>PBNOB</td>
<td></td>
</tr>
<tr>
<td>2SNOM</td>
<td>FSNO</td>
<td>MOPNOA</td>
<td>PBNOE</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input Device Codes</th>
<th>2SSNOL</th>
<th>MOPNOB</th>
<th>PBNOQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>2SSNOR</td>
<td>FTSNNO</td>
<td>MOPNOR</td>
<td>PENC</td>
</tr>
<tr>
<td>3KNCC</td>
<td>FTSSNC</td>
<td>NCTC</td>
<td>PENO</td>
</tr>
<tr>
<td>3KNCL</td>
<td>FTSSNO</td>
<td>NCTO</td>
<td>PRXNC</td>
</tr>
<tr>
<td>3KNCM</td>
<td>LSOC</td>
<td>NOTC</td>
<td>PRXNO</td>
</tr>
<tr>
<td>3KNOR</td>
<td>LSOC</td>
<td>NOTO</td>
<td>PSNC</td>
</tr>
<tr>
<td>3KNOL</td>
<td>LSNC</td>
<td>PBNCB</td>
<td>PSNO</td>
</tr>
<tr>
<td>3KNOM</td>
<td>MCRNC</td>
<td>PBNCG</td>
<td>TSNC</td>
</tr>
<tr>
<td>3KNOR</td>
<td>MCRNO</td>
<td>PBNCK</td>
<td>TSNO</td>
</tr>
</tbody>
</table>

Output Device Codes

<table>
<thead>
<tr>
<th>Output Device Codes</th>
<th>CNCOIL</th>
<th>PLTG</th>
<th>SPOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRCOIL</td>
<td>PLTR</td>
<td>SSOL</td>
<td></td>
</tr>
<tr>
<td>DSOL</td>
<td>PLTW</td>
<td>TMCOIL</td>
<td></td>
</tr>
<tr>
<td>MTLTA</td>
<td>PTPLTA</td>
<td>TROIL</td>
<td></td>
</tr>
<tr>
<td>MTLTB</td>
<td>PTPLTB</td>
<td>TSOL</td>
<td></td>
</tr>
<tr>
<td>MTPG</td>
<td>PTPLTC</td>
<td>WBA</td>
<td></td>
</tr>
<tr>
<td>MTPG</td>
<td>PTPLTG</td>
<td>WBB</td>
<td></td>
</tr>
<tr>
<td>MTPGT</td>
<td>PTPLTR</td>
<td>WBG</td>
<td></td>
</tr>
<tr>
<td>MTPGT</td>
<td>PTPLTW</td>
<td>WBL</td>
<td></td>
</tr>
<tr>
<td>PLTA</td>
<td>RCOIL</td>
<td>WBR</td>
<td></td>
</tr>
<tr>
<td>PLTB</td>
<td>SIZEO</td>
<td>WBZ</td>
<td></td>
</tr>
<tr>
<td>PLTC</td>
<td>SIZER</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.”

```vbnet
Result As Boolean
Result = gAddrSymRecord.SetDeviceCode("SPOT")
```

**SetScope**

Use this method to set the local program file of the AddrSymRecord.

**Syntax**

```vbnet
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.

```vbnet
Result As Boolean
Result = gAddrSymRecord.SetScope(0) 'Makes the gAddrSymRecord global
```

**SetSymbol**

Use this method to set the symbol of an AddrSymRecord.

**Syntax**

```vbnet
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.

```vbnet
Result As Boolean
Result = gAddrSymRecord.SetSymbol("TEST")
```
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.
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.

The following commented code example illustrates how you might get the RungCmntPageTitleRecords collection from the LogixProject object. This example adds error checking and notification.
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**  
This property returns an Application object that represents the RSLogix application.

**Count**  
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**  
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.

```vba
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.

```vba
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 that is attached to the destination record.
DestRungNumber - The rung number that is attached to the destination 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.

```vcl
gRungCommentPageTitleRecord = gRungCommentPageTitleRecords.GetRecordViaFileRung(3,4)
```

**GetRecordViaIndex**

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.

```vcl
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(\textit{Index as Long}, \textit{RungCommentSearchString as String}, \textit{CaseSensitive as Boolean}, \textit{Wrap as Boolean}) as RungCommentPageTitleRecord

Arguments
\textit{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.

\textit{RungCommentSearchString} - The string searched for in the Rung Comment database.

\textit{CaseSensitive} - If set to True, the case of any letters in the RungCommentsSearchString will be used to filter the search.

\textit{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.

\begin{verbatim}
Dim Index As Long
Index = 10
gRungCommentPageTitleRecord =
gRungCommentPageTitleRecords.GetRecordViaRungComment(Index, "PLC-5", False, True)
\end{verbatim}

\begin{itemize}
\item \textbf{RemoveRecordViaAddress} \hspace{1cm} \textbf{Boolean}
\end{itemize}

Use this method to remove a record from the Rung Comment/Page Title record by indicating its address.

Syntax
RemoveRecordViaAddress(\textit{Address as String}) as Boolean

Arguments
\textit{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.

```vba
Dim Res As Boolean
Res = gRungCommentPageTitleRecords.RemoveRecordViaAddress("B3:0");
```

**RemoveRecordViaFileRung**

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.

```vba
Dim Res As Boolean
Res = gRungCommentPageTitleRecords.RemoveRecordViaFileRung(4,2)
```

**RemoveRecordViaIndex**

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.

```vbnet
Dim Index As Long
Index = 10
CommentsReplaced As Long
CommentsReplaced = gRungCommentPageTitleRecords.SearchAndReplacePageTitle (Index, "test", "Debug", False, True, False)
```

---

**SearchAndReplaceRungComment**

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**

```vbnet
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.

- **RungCommentSearchString** - The string that is searched for in the Rung Comment database.

- **RungCommentReplaceString** - 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.
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.

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

```vbs
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
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 `RungCommentPageTitleRecord` object.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Read Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>String</td>
<td>Read Only</td>
</tr>
<tr>
<td>Application</td>
<td>Application</td>
<td>Read Only</td>
</tr>
<tr>
<td>IsAttachedToAddress</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td>PageTitle</td>
<td>String</td>
<td>Read Only</td>
</tr>
<tr>
<td>ProgFile</td>
<td>Long</td>
<td>Read Only</td>
</tr>
<tr>
<td>RungComment</td>
<td>String</td>
<td>Read Only</td>
</tr>
<tr>
<td>RungNumber</td>
<td>Long</td>
<td>Read Only</td>
</tr>
</tbody>
</table>

- **Address**: 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**: This property returns an `Application` object that represents the RSLogix application.

- **IsAttachedToAddress**: 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**: This property returns a string containing the page title of the `RungCommentPageTitleRecord`.

- **ProgFile**: 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**: This property returns a string containing the rung comment of the `RungCommentPageTitleRecord`.

- **RungNumber**: 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**

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**

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 RungComment/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.

```plaintext
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.
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.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>AddNodePrivilegeEntry</td>
<td>-None-</td>
</tr>
<tr>
<td>CurrentClass</td>
<td>ChangeNodePrivilegeInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ClassLogin</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DownloadPrivChanges</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GetChannelPrivileges</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GetDataFilePrivileges</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GetDefaultClass</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GetFeaturePrivileges</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GetNodePrivilegeInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GetProgFilePrivileges</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IsClassPasswordProtected</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RefreshChannelPrivesFromOnline</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RefreshPassPrivesFromOnline</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RemoveNodePrivilegeEntry</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SetChannelPrivileges</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SetClassPassword</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SetDataFilePrivileges</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SetDefaultClass</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SetFeaturePrivileges</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SetProcessorPassword</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SetProgFilePrivileges</td>
<td></td>
</tr>
</tbody>
</table>

PasswordPrivilegeConfig object • 171
The following commented code example illustrates how you might access the PasswordPrivilegeConfig object.

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

- **CurrentClass**
  - Integer - Read Only
  - This property returns an integer value that returns the current class that the project is logged into.

- **NodePrivilegeEntryCount**
  - Application - Read Only
  - 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.

```plaintext
Index As Integer
Index = gPassPriv.AddNodePrivilegeEntry(lgxChan2A, 16, 10, 2)
```

**ChangeNodePrivilegeInfo**

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.

```vbnet
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
```

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.

```vbnet
Rights As lgxPrivilege
Rights = gPassPriv.GetDataFilePrivileges(5, 3)
If Rights <> lgxFailedToGetPriv Then
    If (Rights = lgxNoPriv) Then
        MsgBox("No Privileges")
    ElseIf (Rights = lgxReadPriv) Then
        MsgBox("Read Privileges")
    ElseIf (Rights = lgxWritePriv) Then
        MsgBox("Write Privileges")
    ElseIf (Rights = lgxReadWritePriv) Then
        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.

```vbnet
nClass As Long
nClass = gPassPriv.GetDefaultClass(lgxPLC5_CH0)
```

**GetFeaturePrivileges**
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**

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.

```vbnet
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

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 in 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) Then
    MsgBox("Read Privileges")
ElseIf (Rights = lgxWritePriv) Then
    MsgBox("Write Privileges")
ElseIf (Rights = lgxReadWritePriv) Then
    MsgBox("Read and Write Privileges")
End If

IsClassPasswordProtected

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.

```plaintext
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.

```plaintext
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.

```vbnet
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**

```vbnet
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.

```vbnet
Result As Boolean
Result = gPassPriv.RemoveNodePrivilegeEntry(0)
```

### SetChannelPrivileges Boolean

Use this method to set the specified channel privileges in the current project.

**Syntax**

```vbnet
SetChannelPrivileges(Channel as lgxChannel, Class as Integer, Privilege as lgxPrivilege) as Boolean
```

**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.
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.

```vba
Rights As lgxPrivilege
Rights = lgxWritePriv
gPassPriv.SetChannelPrivileges(lgxPLC5_Ch2, 3, Rights)
```

SetClassPassword

<table>
<thead>
<tr>
<th>Boolean</th>
</tr>
</thead>
</table>

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("oldpassword", "newpass", 1)

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.

```vbnet
Rights As lgxPrivilege
Rights = lgxNoPriv
gPassPriv.SetDataFilePrivileges(7, 2, Rights)
```

**SetDefaultClass**

Use this method to set the default class of the offline editor or any of the processor’s channels.

**Syntax**

```vbnet
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.

```vbnet
Result As Boolean
Result = gPassPriv.SetDefaultClass(lgxPLC5_Ch1A, 2)
```

**SetFeaturePrivileges**

Use this method to set the specified feature privileges in the current project.

**Syntax**

```vbnet
SetFeaturePrivileges(PrivilegeType as lgxPrivilege, Class as Integer, Enabled as Boolean) as Boolean
```

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.
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

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 ‘oldpasswrd’ to ‘newpass.’

Result As Boolean
Result = gPassPriv.SetProcessorPassword("oldpasswrd","newpass")

Syntax
SetProgFilePrivileges(ProgFile as Long, Class as Integer, Privileges as lgxPrivilege) 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

Object model diagrams

Introduction

The following pages illustrate the object models for RSLogix 5 and RSLogix 500 Programming Software.
RSLogix 5 object model summary

**Application**
- Application
- AutoSaveInterval
- BackupCount
- EnableAutoArrange
- EnableAutoSave
- EncodedRouteString
- FullName
- LibrarySearchPath
- MaxDescriptionLineLength
- MaxSymbolLength
- Name
- NumberOfDescriptionLines
- Parent
- PromptForRevNote
- ProVersion
- SourceSearchPath
- VBVersion
- VBE
- Version
- Visible
- WindowHandle
- WindowState
- [FileNew]
- [FileOpen]
- [GetActiveProject]
- [GetProcessorTypes]
- [GoOffline]
- [GoOnline]
- [Quit]
- [Upload]
- <AfterUpload>
- <BeforeFileNew>
- <BeforeFileOpen>
- <BeforeOffline>
- <BeforeOnline>
- <BeforeDownload>
- <BeforeSaveAs>
- <BeforeSave>
- <BeforeClose>
- <BeforeUpload>
- <ClosingAllProjects>
- <Quit>

**LogixProject**
- AddrSymRecords
- Application
- DataFiles
- FullName
- Modified
- Name
- Online
- Parent
- PasswordPrivilegeCfg
- Processor
- ProgramFiles
- ReportOptions
- Revision
- RevisionNotes
- RunGcmnPageTitleRecords
- [Close]
- [DisplayReportOptions]
- [Download]
- [GotoDataFileElement]
- [GotoProgramFile]
- [ImportDataBase]
- [PrintReport]
- [Save]
- [SaveAs]
- [ShowControllerProperties]
- [ShowDataFile]
- [ShowDataTablesProperties]
- [ShowProgramFile]
- [ShowProgramFilesProperties]
- [VerifyProject]
- [Verify ProgramFile]
- <AfterDownload>
- <AfterOpen>
- <AfterSave>
- <AfterClose>
- <BeforeDownload>
- <BeforeSaveAs>
- <BeforeSave>
- <BeforeClose>
- <BeforeUpload>
- <OnlineOfflineFileClosing>

**Processor**
- Application
- CanAssembleEdits
- CanCancelEdits
- CanTestEdits
- CanUnTestEdits
- CurrentPLC5MemSize
- DefaultDriver
- DestNodeOctal
- DriverName
- DriverTimeout
- EditsActive
- EditsPresent
- Emulator
- EncodedRouteString
- Faulted
- HasPasswordPrivileges
- KeySwitchPosition
- Name
- Node
- NumberOfMemSizeChoices
- Online
- OnlineChangesMade
- ProcessorMode
- Revision
- Series
- SubRevision
- Type
- [ClearAllForces]
- [DisableForces]
- [EnableForces]
- [GetPLC5MemSizeByIndex]
- [SetPLC5MemSize]

**DataFiles**
- Application
- [Add]
- [Count]
- [Item]
- [Remove]
- [GetDataValue]
- [SetDataValue]
- <AfterDownload>
- <BeforeDownload>
- <BeforeSaveAs>
- <BeforeSave>
- <BeforeClose>
- <BeforeUpload>
- <OnlineOfflineFileClosing>

**Returns Application Object**
- ReportOptions Object
- RevisionNotes Object
- ProgramFile Object
- LadderFile Object
- DataFile Object

**Note:** The diagram and text provide a visual and textual representation of the RSLogix 5 object model summary. The diagram illustrates the relationships and methods of various objects within the application, data files, and processor domains.
RSLogix 5 Object Model Summary, continued

### DataFile
- Application
- CanBeDeleted
- CanBeMonitored
- CanChangeScope
- CanChangeSize
- Debug
- Description
- FileNumber
- FormattedName
- GlobalScope
- InUse
- LocalScope
- MaxDescriptionLength
- MaxNameLength
- Name
- NumberOfElements
- 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]
- [RefreshChannelPrivesFromOnline]
- [RefreshPassPrivsFromOnline]
- [RemoveNodePrivilegeEntry]
- [SetChannelPrivileges]
- [SetClassPassword]
- [SetDataFilePrivileges]
- [SetDefaultClass]
- [SetFeaturePrivileges]
- [SetProcessorPassword]
- [SetProgFilePrivileges]

### 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]

### RevisionNotes
- Application
- InternalRevision
- Revision
- [Count]
- [RevisionNote]

### RevisionNotes
- Application
- InternalRevision
- Revision
- [Count]
- [RevisionNote]

### Rung
- Active
- Application
- Comment
- DbaseID
- EditsActive
- EndRung
- FileNumber
- Modified
- NumberOfInstructions
- Online
- Output
- RungNumber
- RungType
- RungZoneDisplay
- TempReplace
- Title
- Verified

### KEY
- Property
- [Method]
- <Event>
## RSLogix 5 Object Model Summary, Database Utilities

### AddrSymRecord

<table>
<thead>
<tr>
<th>Property</th>
<th>Method</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Above</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Below</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeviceCode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scope</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SymbolGroup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SetAbove]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SetAddress]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SetBelow]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SetDescription]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SetDeviceCode]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SetScope]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SetSymbol]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SetSymGroup]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### AddrSymRecords

<table>
<thead>
<tr>
<th>Property</th>
<th>Method</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Add]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Duplicate]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[GetRecordIndexViaAddrOrSym]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[GetRecordViaAddrOrSym]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[GetRecordViaDesc]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[GetRecordViaIndex]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[RemoveRecordViaAddrOrSym]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[RemoveRecordViaIndex]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SearchAndReplaceDesc]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### RungCmntPageTitleRecord

<table>
<thead>
<tr>
<th>Property</th>
<th>Method</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsAttachedToAddress</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PageTitle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProgFile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RungComment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RungNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SetAddress]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SetPageTitle]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SetProgFileAndRung]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SetRungComment]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### RungCmntPageTitleRecords

<table>
<thead>
<tr>
<th>Property</th>
<th>Method</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[AddRecordAttachedToProgFileAndRung]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[AddRecordAttachedToAddress]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[DuplicateViaAddress]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[DuplicateViaFileRung]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[GetRecordViaAddrOrSym]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[GetRecordViaIndex]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[RemoveRecordViaAddrOrSym]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[RemoveRecordViaFileRung]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SearchAndReplacePageTitle]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SearchAndReplaceRungComment]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### KEY

- **Object**
- **Property**
- **Method**
- **<Event>**
RSLogix 500 object model summary

**Application**
- Application
- AutoSaveInterval
- BackupCount
- EnableAutoArrange
- EnableAutoSave
- EncodedRouteString
- FullName
- LibrarySearchPath
- MaxDescriptionLineLength
- MaxSymbolLength
- Name
- NumberOfDescriptionLines
- Parent
- PromptForRevNote
- ProVersion
- SourceSearchPath
- VBAVersion
- VBE
- Version
- Visible
- WindowHandle
- WindowState

- [FileNew]
- [FileOpen]
- [GetActiveProject ]
- [GetProcessorTypes]
- [GoOffline]
- [GoOnline]
- [Quit]
- [Upload]

- <AfterUpload>
- <BeforeFileNew>
- <BeforeFileOpen>
- <BeforeOffline>
- <BeforeOnline>
- <BeforeUpload>
- <ClosingAllProjects>
- <Quit>

- **Returns Application Object**

**LogixProject**
- AddrSymRecords
- Application**
- DataFiles
- FullName
- Modified
- Name
- Online
- Parent
- Processor
- ProgramFiles
- ReportOptions
- Revision
- RevisionNotes
- RungCmntPageTitleRecords

- [Close]
- [DisplayReportOptions]
- [Download]
- [GotoDataFileElement ]
- [GotoProgramFile]
- [ImportDataBase]
- [PrintReport ]
- [Save]
- [SaveAs]
- [ShowControllerProperties]
- [ShowDataFile]
- [ShowDataTablesProperties]
- [ShowProgramFile]
- [ShowProgramFilesProperties]
- [VerifyProject ]
- [VerifyProgramFile]

- <AfterDownload>
- <BeforeOpen>
- <BeforeSave>
- <BeforeClose>
- <BeforeDownload>
- <BeforeSave>
- <BeforeSaveAs>
- <FinishedReport>
- <FinishedVerify>
- <OnlineOfflineFileClosing>

**Processor**
- Application**
- CanAssembleEdits
- CanCancelEdits
- CanTestEdits
- CanUntestEdits
- DefaultDriver
- DestNodeOctal
- DriverName
- DriverTimeout
- EditsActive
- EditsPresent
- Emulator
- EncodedRouteString
- Faulted
- KeySwitchPosition
- Name
- Node
- Online
- OnlineChangesMade
- ProcessorMode
- ProgramID
- Type

- [ClearAllForces]
- [DisableForces]
- [EnableForces]

**DataFiles**
- Application**

- [Add ]
- [Count]
- [GetDataValue]
- [Item]
- [Remove]
- [SetDataValue]

**Returns Application Object**

- ReportOptions Object
- RevisionNotes Object
- ProgramFile Object
- LadderFile Object
- DataFile Object
RSLogix 500 Object Model Summary, continued

### DataFile
- Application**
- CanBeDeleted
- CanBeMonitored
- ChangeScope
- ChangeSize
- 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

### RevisionNotes
- Application**
- InternalRevision
- Revision
- [Count]
- [RevisionNote]

**KEY**
- Property
- [Method]
- <Event>
RSLogix 500 object model summary, database utilities

<table>
<thead>
<tr>
<th>AddrSymRecord</th>
<th>AddrSymRecords</th>
<th>RungCmntPageTitleRecords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Above</td>
<td>Application**</td>
<td>Application**</td>
</tr>
<tr>
<td>Address</td>
<td>Count</td>
<td>Count</td>
</tr>
<tr>
<td>Application**</td>
<td>[Add]</td>
<td>[AddRecordAttachedToProgFileAndRung]</td>
</tr>
<tr>
<td>Below</td>
<td>[Duplicate]</td>
<td>[AddRecordAttachedToAddress]</td>
</tr>
<tr>
<td>Description</td>
<td>[GetRecordIndexViaAddrOrSym]</td>
<td>[DuplicateViaAddress]</td>
</tr>
<tr>
<td>DeviceCode</td>
<td>[GetRecordViaAddrOrSym]</td>
<td>[DuplicateViaFileRung]</td>
</tr>
<tr>
<td>Scope</td>
<td>[GetRecordViaDesc]</td>
<td>[GetRecordViaAddress]</td>
</tr>
<tr>
<td>Symbol</td>
<td>[GetRecordViaIndex]</td>
<td>[GetRecordViaFileRung]</td>
</tr>
<tr>
<td>SymbolGroup</td>
<td>[RemoveRecordViaAddrOrSym]</td>
<td>[GetRecordViaIndex]</td>
</tr>
<tr>
<td></td>
<td>[SearchAndReplaceDesc]</td>
<td>[GetRecordViaPageTitle]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[GetRecordViaRungComment]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[RemoveRecordViaAddress]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[RemoveRecordViaFileRung]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[RemoveRecordViaIndex]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[SearchAndReplacePageTitle]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[SearchAndReplaceRungComment]</td>
</tr>
</tbody>
</table>

**KEY**

- **Object**
  - Property
  - [Method]
  - <Event>
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.
## lgxDataFileTypeConstants (RSLogix 5)

Used with the DataFile object and DataFiles collection. Not all may apply.

<table>
<thead>
<tr>
<th>Value</th>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>lgxDTUNKNOWN</td>
<td>The type of data file is not recognized.</td>
</tr>
<tr>
<td>0</td>
<td>lgxDTBINARY</td>
<td>Indicates a Binary data file type.</td>
</tr>
<tr>
<td>1</td>
<td>lgxINTEGER</td>
<td>Indicates an Integer data file type.</td>
</tr>
<tr>
<td>2</td>
<td>lgxDTEXPANDER</td>
<td>Indicates an Expander data file type.</td>
</tr>
<tr>
<td>3</td>
<td>lgxDTASCII</td>
<td>Indicates the ASCII data file type.</td>
</tr>
<tr>
<td>4</td>
<td>lgxDTBCD</td>
<td>Indicates the Binary Coded Decimal data file type.</td>
</tr>
<tr>
<td>5</td>
<td>lgxDTSFSTATUS</td>
<td>Indicates an SFC status data file type.</td>
</tr>
<tr>
<td>6</td>
<td>lgxDTSTRING</td>
<td>Indicates a String data file type.</td>
</tr>
<tr>
<td>7</td>
<td>lgxDTBLOCKXFER</td>
<td>Indicates a Block Transfer data file type.</td>
</tr>
<tr>
<td>8</td>
<td>lgxDTCONTROLNET</td>
<td>Indicates a ControlNet data file type.</td>
</tr>
<tr>
<td>9</td>
<td>lgxDTTIMER</td>
<td>Indicates a Timer data file type.</td>
</tr>
<tr>
<td>10</td>
<td>lgxDT COUNTER</td>
<td>Indicates a Counter data file type.</td>
</tr>
<tr>
<td>11</td>
<td>lgxDTCONTROL</td>
<td>Indicates a Control data file type.</td>
</tr>
<tr>
<td>12</td>
<td>lgxDTTOKENENDATA</td>
<td>Indicates a Token passing data file type.</td>
</tr>
<tr>
<td>13</td>
<td>lgxDT FLOAT</td>
<td>Indicates a Floating Point data file type.</td>
</tr>
<tr>
<td>14</td>
<td>lgxDTMESSAGE</td>
<td>Indicates a Message data file type.</td>
</tr>
<tr>
<td>15</td>
<td>lgxDTPIDBLOCK</td>
<td>Indicates a PID Block data file type.</td>
</tr>
<tr>
<td>16</td>
<td>lgxDTLONG</td>
<td>Indicates a Long data file type.</td>
</tr>
<tr>
<td>17</td>
<td>lgxDTMUTEX</td>
<td>A Mutex file type, handshaking between 2+ async threads.</td>
</tr>
<tr>
<td>18</td>
<td>lgxDTEVENT</td>
<td>Indicates an Event data file type.</td>
</tr>
<tr>
<td>19</td>
<td>lgxDTMANUALEVENT</td>
<td>Indicates a Manual Event data file type.</td>
</tr>
<tr>
<td>20</td>
<td>lgxDTDOUBLE</td>
<td>Indicates a Double data file type.</td>
</tr>
<tr>
<td>21</td>
<td>lgxDT TIME</td>
<td>A SoftLogix specific data type representing current date and time.</td>
</tr>
<tr>
<td>22</td>
<td>lgxDTINT64</td>
<td>Indicates a 64-bit Integer data file type.</td>
</tr>
<tr>
<td>23</td>
<td>lgxDTUNUSED</td>
<td>Indicates an unused data file.</td>
</tr>
<tr>
<td>24</td>
<td>lgxDTOUTPUT</td>
<td>Indicates an Output data file type.</td>
</tr>
<tr>
<td>25</td>
<td>lgxDTINPUT</td>
<td>Indicates an Input data file type.</td>
</tr>
<tr>
<td>26</td>
<td>lgxDTSTATUS</td>
<td>Indicates a status data file type.</td>
</tr>
<tr>
<td>27</td>
<td>lgxDTM0</td>
<td>An M0 file type. (Controls operation of devices on RIO link.)</td>
</tr>
<tr>
<td>28</td>
<td>lgxDTM1</td>
<td>An M1 file type. (Status of devices on RIO link.)</td>
</tr>
<tr>
<td>29</td>
<td>lgxDTSO</td>
<td>Indicates a SLC I/O data file.</td>
</tr>
<tr>
<td>30</td>
<td>lgxDTSI</td>
<td>Indicates a SLC I/O data file.</td>
</tr>
<tr>
<td>31</td>
<td>lgxLABEL</td>
<td>Indicates a Label file.</td>
</tr>
<tr>
<td>32</td>
<td>lgxDTSBR</td>
<td>Indicates a subroutine.</td>
</tr>
<tr>
<td>33</td>
<td>lgxDTRET</td>
<td>Not useful to object model. Used internally only.</td>
</tr>
<tr>
<td>34</td>
<td>lgxDTBOOL</td>
<td>Indicates a Boolean data file type.</td>
</tr>
<tr>
<td>35</td>
<td>lgxDTNUMBEROFDATATYPES</td>
<td>Not useful to object model. Used internally only.</td>
</tr>
<tr>
<td>36</td>
<td>lgxDTRESERVED1</td>
<td>Reserved</td>
</tr>
<tr>
<td>37</td>
<td>lgxDTRESERVED2</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xff</td>
<td>lgxDTSYSTEMTYPES</td>
<td>Not useful to object model. Used internally only.</td>
</tr>
<tr>
<td>0xfff</td>
<td>lgxDTMAXSYSTEMTYPE</td>
<td>Not useful to object model. Used internally only.</td>
</tr>
<tr>
<td>32768</td>
<td>lgxUSERTYPE</td>
<td>Not useful to object model. Used internally only.</td>
</tr>
</tbody>
</table>
**lgxDataFileTypeConstants (RSLogix 500)**

Used with the **DataFile** object and **DataFiles** collection. Not all may apply.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>lgxDTUNKNOWED      The type of data file is not recognized.</td>
</tr>
<tr>
<td>0</td>
<td>lgxDTOUTPUT        Indicates an Output data file type.</td>
</tr>
<tr>
<td>1</td>
<td>lgxDTINPUT         Indicates an Input data file type.</td>
</tr>
<tr>
<td>2</td>
<td>lgxDTSTATUS        Indicates a status data file type.</td>
</tr>
<tr>
<td>3</td>
<td>lgxDTBINARY        Indicates a Binary data file type.</td>
</tr>
<tr>
<td>4</td>
<td>lgxDTTIMER         Indicates a Timer data file type.</td>
</tr>
<tr>
<td>5</td>
<td>lgxDTCOUNTER       Indicates a Counter data file type.</td>
</tr>
<tr>
<td>6</td>
<td>lgxDTCONTROL       Indicates a Control data file type.</td>
</tr>
<tr>
<td>7</td>
<td>lgxINTEGER         Indicates an Integer data file type.</td>
</tr>
<tr>
<td>8</td>
<td>lgxDTFLOAT         Indicates a Floating Point data file type.</td>
</tr>
<tr>
<td>9</td>
<td>lgxDTRESERVED1     Reserved</td>
</tr>
<tr>
<td>10</td>
<td>lgxDTUNUSED       Indicates an unused data file.</td>
</tr>
<tr>
<td>11</td>
<td>lgxDTRESERVED2     Reserved</td>
</tr>
<tr>
<td>12</td>
<td>lgxDTM1            An M1 file type (Status of devices on RIO link).</td>
</tr>
<tr>
<td>13</td>
<td>lgxDTM0            An M0 file type. (Controls operation of devices on RIO link.)</td>
</tr>
<tr>
<td>14</td>
<td>lgxDTSTRING        Indicates a String data file type.</td>
</tr>
<tr>
<td>15</td>
<td>lgxDTASCII         Indicates the ASCII data file type.</td>
</tr>
<tr>
<td>16</td>
<td>lgxDTLONG          Indicates a Long data file type.</td>
</tr>
<tr>
<td>17</td>
<td>lgxDTHSCOUNTER     Indicates a high speed counter data type.</td>
</tr>
<tr>
<td>18</td>
<td>lgxDTPULESE_TRAIN_OUT Indicates a pulse train output data type.</td>
</tr>
<tr>
<td>19</td>
<td>lgxDTMESSAGE       Indicates a Message data file type.</td>
</tr>
<tr>
<td>20</td>
<td>lgxDTEVENT_INPUT_INT Indicates an event input interrupt.</td>
</tr>
<tr>
<td>21</td>
<td>lgxDTREAL_TIME_CLOCK Indicates a real time clock data type.</td>
</tr>
<tr>
<td>22</td>
<td>lgxDBASE_HARDWARE_INFO Indicates the BHI (Base Hardware Information) function file.</td>
</tr>
<tr>
<td>23</td>
<td>lgxDTMEM_MODULE_INFO Indicates the MMI (Memory Module Information) function file.</td>
</tr>
<tr>
<td>24</td>
<td>lgxDTDATA_ACCESS_TERM_INFO Indicates the DAT (Data Access Terminal) function file.</td>
</tr>
<tr>
<td>25</td>
<td>lgxDTTPI_INT        Indicates a selectable timed interrupt.</td>
</tr>
<tr>
<td>26</td>
<td>lgxDTCOM_STATUS     Indicates a communications status data file type.</td>
</tr>
<tr>
<td>27</td>
<td>lgxDTIOMOD_STATUS   Indicates an I/O module status data file type.</td>
</tr>
<tr>
<td>28</td>
<td>lgxDTWIDTH_MOD      Indicates a pulse-width module data file type.</td>
</tr>
<tr>
<td>29</td>
<td>lgxDTDATA_LOG_STATUS Indicates a data log status data file type.</td>
</tr>
<tr>
<td>30</td>
<td>lgxDTPLS            Indicates a Programmable Limit Switch data file type.</td>
</tr>
<tr>
<td>31</td>
<td>lgxDTBCD            Indicates the Binary Coded Decimal data file type.</td>
</tr>
<tr>
<td>32</td>
<td>lgxDTPBKER          Indicates a Block Transfer data file type.</td>
</tr>
<tr>
<td>33</td>
<td>lgxDTBFC_STATUS     Indicates an SFC status data file type.</td>
</tr>
<tr>
<td>34</td>
<td>lgxDTTOKENDATA      Indicates a Token passing data file type.</td>
</tr>
<tr>
<td>35</td>
<td>lgxDTCONTROLNET    Indicates a ControlNet data file type.</td>
</tr>
<tr>
<td>36</td>
<td>lgxLABEL           Indicates a Label file.</td>
</tr>
<tr>
<td>37</td>
<td>lgxDTSBR            Indicates a subroutine.</td>
</tr>
<tr>
<td>38</td>
<td>lgxDTRET            Not useful to object model. Used internally only.</td>
</tr>
<tr>
<td>39</td>
<td>lgxDTBOOL          Indicates a Boolean data file type.</td>
</tr>
<tr>
<td>32768</td>
<td>lgxDTUSERTYPE0     Not useful to object model. Used internally only.</td>
</tr>
</tbody>
</table>
lgxKeyPositionConstants (RSLogix 5 and 500)

Used with the Processor object.

0  lgxUnknownKey  Processor keyswitch position is not known.
1  lgxKeyRemote   Processor keyswitch in Remote position.
2  lgxKeyProgram  Processor keyswitch in Program position.
3  lgxKeyRun      Processor keyswitch in Run position.

lgxOnlineAction (RSLogix 5 and 500)

Used with the Application and LogixProject objects.

1  lgxGoOnline     Instructs application to go online with the processor.
2  lgxGoOffline    Instructs application to go offline with the processor.
**lgxProcessorTypeConstants (RSLogix 5)**

Used with the Application and Processor objects.

-1  lgxLUNKOWNPROC
1   lgxPLC_515
2   lgxPLC_512
3   lgxPLC_5VME
4   lgxPLC_525
5   lgxPLC_510
6   lgxPLC_540
7   lgxPLC_560
8   lgxPLC_540L
9   lgxPLC_560L
10  lgxPLC_530
11  lgxPLC_511
12  lgxPLC_520
13  lgxPLC_540VME
14  lgxPLC_540VMEL
15  lgxPLC_520E
16  lgxPLC_540E
17  lgxPLC_580
18  lgxPLC_516
19  lgxPLC_526
20  lgxPLC_536
21  lgxPLC_546
22  lgxPLC_546L
23  lgxPLC_566
24  lgxPLC_566L
25  lgxPLC_586
26  lgxPLC_580E
27  lgxPLC_530VME
28  lgxPLC_580VME
29  lgxPLC_520C
30  lgxPLC_540C
31  lgxPLC_560C
32  lgxPLC_580C
33  lgxPLC_520C2
34  lgxPLC_540C2
35  lgxPLC_560C2
36  lgxPLC_580C2
37  lgxPLC_526C2
38  lgxPLC_546C2
39  lgxSOFTLOGIX_5
40  lgxPLC_580VMEL
**lgxProcessorTypeConstants (RSLogix 500)**

Used with the **Application** and **Processor** objects.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>lgxLUNKNOWNPROC</td>
</tr>
<tr>
<td>0</td>
<td>lgx1747_L20A</td>
</tr>
<tr>
<td>1</td>
<td>lgx1747_L20B</td>
</tr>
<tr>
<td>2</td>
<td>lgx1747_L20C_F</td>
</tr>
<tr>
<td>3</td>
<td>lgx1747_L20D</td>
</tr>
<tr>
<td>4</td>
<td>lgx1747_L20E_G</td>
</tr>
<tr>
<td>5</td>
<td>lgx1747_L20F_N</td>
</tr>
<tr>
<td>6</td>
<td>lgx1747_L20G</td>
</tr>
<tr>
<td>7</td>
<td>lgx1747_L20H</td>
</tr>
<tr>
<td>8</td>
<td>lgx1747_L30A</td>
</tr>
<tr>
<td>9</td>
<td>lgx1747_L30B</td>
</tr>
<tr>
<td>10</td>
<td>lgx1747_L30C</td>
</tr>
<tr>
<td>11</td>
<td>lgx1747_L30D</td>
</tr>
<tr>
<td>12</td>
<td>lgx1747_L30E</td>
</tr>
<tr>
<td>13</td>
<td>lgx1747_L30F</td>
</tr>
<tr>
<td>14</td>
<td>lgx1747_L40A</td>
</tr>
<tr>
<td>15</td>
<td>lgx1747_L40B</td>
</tr>
<tr>
<td>16</td>
<td>lgx1747_L40C_F</td>
</tr>
<tr>
<td>17</td>
<td>lgx1747_L40D</td>
</tr>
<tr>
<td>18</td>
<td>lgx1747_L40E</td>
</tr>
<tr>
<td>19</td>
<td>lgx1747_L40F</td>
</tr>
<tr>
<td>20</td>
<td>lgx1747_L511</td>
</tr>
<tr>
<td>21</td>
<td>lgx1747_L514</td>
</tr>
<tr>
<td>22</td>
<td>lgx1747_L524</td>
</tr>
<tr>
<td>23</td>
<td>lgx1747_L532</td>
</tr>
<tr>
<td>24</td>
<td>lgxMICRO</td>
</tr>
<tr>
<td>25</td>
<td>lgx1747_L532B</td>
</tr>
<tr>
<td>26</td>
<td>lgx1747_L542A</td>
</tr>
<tr>
<td>27</td>
<td>lgx1747_L542C</td>
</tr>
<tr>
<td>28</td>
<td>lgx1747_L542D</td>
</tr>
<tr>
<td>29</td>
<td>lgx1747_L543</td>
</tr>
<tr>
<td>30</td>
<td>lgxMICRO_DH485</td>
</tr>
<tr>
<td>31</td>
<td>lgx1747_L551</td>
</tr>
<tr>
<td>32</td>
<td>lgx1747_L552</td>
</tr>
<tr>
<td>33</td>
<td>lgx1747_L553</td>
</tr>
<tr>
<td>34</td>
<td>lgx1747_L554</td>
</tr>
<tr>
<td>35</td>
<td>lgx1747_L555</td>
</tr>
<tr>
<td>36</td>
<td>lgx1747_L556</td>
</tr>
<tr>
<td>37</td>
<td>lgx1747_L557</td>
</tr>
<tr>
<td>38</td>
<td>lgx1747_L558</td>
</tr>
<tr>
<td>39</td>
<td>lgx1747_L559</td>
</tr>
<tr>
<td>40</td>
<td>lgx1747_L561</td>
</tr>
<tr>
<td>41</td>
<td>lgx1747_L562</td>
</tr>
<tr>
<td>42</td>
<td>lgx1747_L563</td>
</tr>
<tr>
<td>43</td>
<td>lgx1747_L564</td>
</tr>
<tr>
<td>44</td>
<td>lgx1747_L565</td>
</tr>
<tr>
<td>45</td>
<td>lgxMICRO_ANALOG</td>
</tr>
<tr>
<td>46</td>
<td>lgx1747_L566</td>
</tr>
<tr>
<td>47</td>
<td>lgx1747_L567</td>
</tr>
<tr>
<td>48</td>
<td>lgx1747_L568</td>
</tr>
<tr>
<td>49</td>
<td>lgxMICRO1500LSP_A</td>
</tr>
<tr>
<td>50</td>
<td>lgxMICRO1500LSP_B</td>
</tr>
<tr>
<td>51</td>
<td>lgxMICRO1500LSP_C</td>
</tr>
<tr>
<td>52</td>
<td>lgxMICRO1500LRP_B</td>
</tr>
<tr>
<td>53</td>
<td>lgxMICRO1500LRP_C</td>
</tr>
<tr>
<td>54</td>
<td>lgxMICRO1500LSP_D</td>
</tr>
<tr>
<td>55</td>
<td>lgxMICRO1500LRP_E</td>
</tr>
<tr>
<td>56</td>
<td>lgxMICRO1500LSP_F</td>
</tr>
<tr>
<td>57</td>
<td>lgxMICRO1500LSP_G</td>
</tr>
<tr>
<td>154</td>
<td>lgx1747_L531E</td>
</tr>
<tr>
<td>155</td>
<td>lgx1747_L531F</td>
</tr>
<tr>
<td>156</td>
<td>lgx1747_L531G</td>
</tr>
<tr>
<td>157</td>
<td>lgx1747_L531H</td>
</tr>
</tbody>
</table>
**lgxProcOnlineState (RSLogix 5)**

Used with the Processor and LogixProject objects. Not all apply. Refer to the chapter information for your specific use.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>lgxOFFLINE</td>
</tr>
<tr>
<td>1</td>
<td>lgxDOWNLOAD</td>
</tr>
<tr>
<td>2</td>
<td>lgxFAULTED</td>
</tr>
<tr>
<td>3</td>
<td>lgxHARDPROGRAM</td>
</tr>
<tr>
<td>4</td>
<td>lgxHARDTEST</td>
</tr>
<tr>
<td>5</td>
<td>lgxHARDRUN</td>
</tr>
<tr>
<td>6</td>
<td>lgxREMOTEPROG</td>
</tr>
<tr>
<td>7</td>
<td>lgxREMOTETEST</td>
</tr>
<tr>
<td>8</td>
<td>lgxREMOTERUN</td>
</tr>
<tr>
<td>9</td>
<td>lgxBREAKPOINTSTOPPED</td>
</tr>
</tbody>
</table>

- Processor mode is offline.
- Processor mode is download.
- Processor mode is faulted.
- Processor mode is hard program.
- Processor mode is hard test.
- Processor mode is hard run.
- Processor mode is remote program.
- Processor mode is remote test.
- Processor mode is remote run.
- Processor mode is breakpoint stopped.

**lgxProcOnlineState (RSLogix 500)**

Used with the Processor and LogixProject objects. Not all apply. Refer to the chapter information for your specific use.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>lgxOFFLINE</td>
</tr>
<tr>
<td>1</td>
<td>lgxDOWNLOAD</td>
</tr>
<tr>
<td>2</td>
<td>lgxREMOTEPROG</td>
</tr>
<tr>
<td>3</td>
<td>lgxSUSPEND</td>
</tr>
<tr>
<td>4</td>
<td>lgxREMOTERUN</td>
</tr>
<tr>
<td>5</td>
<td>lgxTESTCONTINUOUS</td>
</tr>
<tr>
<td>6</td>
<td>lgxTESTSINGLESINGLESCAN</td>
</tr>
<tr>
<td>7</td>
<td>lgxTESTSTEPNOTRUNNING</td>
</tr>
<tr>
<td>8</td>
<td>lgxTESTSTEPRUNNING</td>
</tr>
<tr>
<td>9</td>
<td>lgxHARDDOWNLOAD</td>
</tr>
<tr>
<td>10</td>
<td>lgxHARDPROGRAM</td>
</tr>
<tr>
<td>11</td>
<td>lgxHARDSUSPEND</td>
</tr>
<tr>
<td>12</td>
<td>lgxHARDRUN</td>
</tr>
<tr>
<td>13</td>
<td>lgxFAULTED</td>
</tr>
</tbody>
</table>

- Processor mode is offline.
- Processor mode is download.
- Processor mode is remote program.
- Suspend instruction executed while in remote program mode.
- Processor mode is remote run.
- Processor is in continuous test mode.
- Processor is in single scan test mode.
- Processor test step mode is not running.
- Processor test step mode is running.
- Processor mode is hard download.
- Processor mode is hard program.
- Suspend instruction executed while in hard program mode.
- Processor mode is hard run.
- Processor mode is faulted.
lgxProgramFileTypeConstants (RSLogix 5)

Used with the ProgramFile object and ProgramFiles collections. Not all apply. Refer to the chapter information for your specific use.

- 0 lgxHEADER: Indicates a header program file type.
- 1 lgxLADDER: Indicates a ladder program file type.
- 2 lgxSFCNEW: Indicates a new sequential function chart program file type.
- 3 lgxSFCOLD: Indicates an old sequential function chart program file type.
- 4 lgxSTX: Indicates a structured text program file type.
- 5 lgxIOFILE: Indicates an input/output program file type.
- 6 lgxSFCRTRL: Not useful to object model. Used internally only.
- 7 lgxPLC_2: Indicates a PLC-2 program file type.
- 8 lgxCONFIG: Indicates a configuration file type.
- 9 lgxCAR: Indicates a CAR program file type.
- 10 lgxCOPROC: Not useful to object model. Used internally only.
- 11 lgxUNUSED: Indicates an unused program file.

lgxProgramFileTypeConstants (RSLogix 500)

Used with the ProgramFile object and ProgramFiles collections. Not all apply. Refer to the chapter information for your specific use.

- 0 lgxHEADER: Indicates a header program file type.
- 1 lgxLADDER: Indicates a ladder program file type.
- 2 lgxSFC: Indicates a sequential function chart program file type.
- 3 lgxUNUSED: Indicates an unused program file.

lgxRungZoneTypes (RSLogix 5 and 500)

Used with the Rung object.

- 0 lgxPlainRung: Indicates a plain, unedited rung.
- 1 lgxReplaceRung: Indicates a replaced rung.
- 2 lgxInsertRung: Indicates an inserted rung.
- 3 lgxDeleteRung: Indicates a deleted rung.
- 4 lgxEditRung: Indicates an edited rung.
- 5 lgxTmpInsertRung: Indicates a temporary inserted rung.
- 6 lgxTmpReplaceRung: Indicates a temporary replacement rung.
- 7 lgxAnyRDRung: Indicates any type of rung.

lgxSaveAction (RSLogix 5 and 500)

Used with the LogixProject object. The only valid selections appear in the table below.

- 0 lgxNoAction: Indicates no external database files saved.
- 1 lgxSaveNativeExternalDB: Indicates database save to Native External file format.
- 2 lgxSaveAlExternalDB: Indicates database save to Al External file format.
- 3 lgxSaveAPSExternalDB: Indicates database save to APS External file format.
**lgxUpDownloadAction (RSLogix 5 and 500)**

Used with the Application object.

1  lgxUploadCreateNew  Upload and create new project.
2  lgxUploadCurrent   Upload and use the current project.
3  lgxUploadPath     Upload and use the project specified at path indicated.

**lgxWindowStateConstants (RSLogix 5 and 500)**

Used with the Application object.

0  lgxWindowStateNormal  Show application in normal window.
1  lgxWindowStateMinimized Application is minimized to an icon.
2  lgxWindowStateMaximized Application is maximized to full screen.

**lgxImportDBTypes (RSLogix 5 and 500)**

Used with the LogixProject object.

0  lgxImportAddrSymDB Import the address/symbol database.

**lgxBinary (RSLogix 5)**

Used with the PasswordPrivilegeCfg object.

0  lgxEnabled
1  lgxDisabled
2  lgxInvalid

**lgxChannel (RSLogix 5)**

Used with the PasswordPrivilegeCfg object.

0  lgxPLCS_Ch0 Channel 0
1  lgxPLCS_Ch1A Channel 1A
2  lgxPLCS_Ch1B Channel 1B
3  lgxPLCS_Ch2A Channel 2A
4  lgxPLCS_Ch2B Channel 2B
5  lgxPLCS_Ch2 Channel 2
6  lgxPLCS_Ch3A Channel 3A
7  lgxPLCS_Offline Offline
**lgxPrivilege (RSLogix 5)**

Used with the **PasswordPrivilegeCfg** object.

- 0  lgxNoPriv
- 1  lgxReadPriv
- 2  lgxWritePriv
- 3  lgxReadWritePriv
- 4  lgxFailedToGetPriv

**lgxPrivilegeType (RSLogix 5)**

Used with the **PasswordPrivilegeCfg** object.

- 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
**lgxErrorTypes (RSLogix 5 and 500)**

Refer to Appendix C for complete information about how you might use the lgxErrorType definition for error handling.

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Error Type Definition</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2147220981</td>
<td>lgxError_Abort_Download</td>
<td>8004020B</td>
</tr>
<tr>
<td>-2147220957</td>
<td>lgxError_ACH_Load_Failed</td>
<td>80040223</td>
</tr>
<tr>
<td>-2147220964</td>
<td>lgxError_Autoconfig_Read_Failed</td>
<td>8004021C</td>
</tr>
<tr>
<td>-2147220923</td>
<td>lgxError_Automation_Inhibited</td>
<td>80040245</td>
</tr>
<tr>
<td>-2147220969</td>
<td>lgxError_Backup_Exists</td>
<td>80040217</td>
</tr>
<tr>
<td>-2147220934</td>
<td>lgxError_BOOTP_Cycle_Power</td>
<td>8004023A</td>
</tr>
<tr>
<td>-2147220991</td>
<td>lgxError_Cancel_Delete_All</td>
<td>80040201</td>
</tr>
<tr>
<td>-2147220966</td>
<td>lgxError_Cannot_Connect_To_Proc</td>
<td>8004021A</td>
</tr>
<tr>
<td>-2147220918</td>
<td>lgxError_Cannot_Find_Activation</td>
<td>8004024A</td>
</tr>
<tr>
<td>-2147220978</td>
<td>lgxError_Comm_Problem</td>
<td>8004020E</td>
</tr>
<tr>
<td>-2147220984</td>
<td>lgxError_Comms_Not_Set</td>
<td>80040208</td>
</tr>
<tr>
<td>-2147220932</td>
<td>lgxError_DataTable_Upload_Failed</td>
<td>8004023C</td>
</tr>
<tr>
<td>-2147220961</td>
<td>lgxDB_Create_Failed</td>
<td>8004021F</td>
</tr>
<tr>
<td>-2147220971</td>
<td>lgxError_DB_Error</td>
<td>80040215</td>
</tr>
<tr>
<td>-2147220942</td>
<td>lgxError_Default_Program</td>
<td>80040232</td>
</tr>
<tr>
<td>-2147220977</td>
<td>lgxError.Download_Failed</td>
<td>8004020F</td>
</tr>
<tr>
<td>-2147220975</td>
<td>lgxError_Enable_Forces_Failed</td>
<td>80040211</td>
</tr>
<tr>
<td>-2147220970</td>
<td>lgxError.Failed_Create_DB</td>
<td>80040216</td>
</tr>
<tr>
<td>-2147220973</td>
<td>lgxError.Failed_Del.Temp_DB</td>
<td>80040213</td>
</tr>
<tr>
<td>-2147220972</td>
<td>lgxError.Failed_Init_DB</td>
<td>80040214</td>
</tr>
<tr>
<td>-2147220968</td>
<td>lgxError.Failed.Online</td>
<td>80040218</td>
</tr>
<tr>
<td>-2147220924</td>
<td>lgxError.File.Is_Read_Only</td>
<td>80040244</td>
</tr>
<tr>
<td>-2147220976</td>
<td>lgxError.Forces.Exist</td>
<td>80040210</td>
</tr>
<tr>
<td>-2147220945</td>
<td>lgxError.General.IO_Error</td>
<td>8004022F</td>
</tr>
<tr>
<td>-2147220979</td>
<td>lgxError.Incompatible.Download_Types</td>
<td>8004020D</td>
</tr>
<tr>
<td>-2147220935</td>
<td>lgxError.Incorrect.Class_Priv</td>
<td>80040239</td>
</tr>
<tr>
<td>-2147220920</td>
<td>lgxError.Invalid_Argument</td>
<td>80040248</td>
</tr>
<tr>
<td>-2147220917</td>
<td>lgxError.Invalid.Class</td>
<td>8004024B</td>
</tr>
<tr>
<td>-2147220922</td>
<td>lgxError.Invalid.Data_File_Type</td>
<td>80040246</td>
</tr>
<tr>
<td>-2147220926</td>
<td>lgxError.Invalidator.During.Compare</td>
<td>80040242</td>
</tr>
<tr>
<td>-2147220987</td>
<td>lgxError.Invalidator.File.Extension</td>
<td>80040205</td>
</tr>
<tr>
<td>-2147220919</td>
<td>lgxError.Invalidator.File.Size</td>
<td>80040249</td>
</tr>
<tr>
<td>-2147220967</td>
<td>lgxError.Invalidator.IO.Object</td>
<td>80040219</td>
</tr>
<tr>
<td>-2147220925</td>
<td>lgxError.Invalidator.Path.Specified</td>
<td>80040243</td>
</tr>
<tr>
<td>-2147220963</td>
<td>lgxError.Invalidator.Rack</td>
<td>8004021D</td>
</tr>
<tr>
<td>-2147220958</td>
<td>lgxError.Invalidator.RSS_File</td>
<td>80040222</td>
</tr>
</tbody>
</table>

continued on following page...
...continued (lgxErrorTypes)
-2147220921  lgxError_Invalid_Secure_Proc_Operation  80040247
-2147220955  lgxError_Invalid_SLC_File  80040225
-2147220952  lgxError_Library_Failure  80040228
-2147220956  lgxError_Library_Load_Failed  80040224
-2147220953  lgxError_Library_Partial_Load_Failed  80040227
-2147220954  lgxError_Library_Warnings_Exist  80040226
-2147220939  lgxError_Multiple_Files_Found  80040235
-2147220929  lgxError_Must_Be_Offline  8004023F
-2147220965  lgxError_No_AutoConfig  8004021B
-2147220928  lgxError_No_Checksum  80040240
-2147220982  lgxError_No_Controller_Response  8004020A
-2147220988  lgxError_No_Future_Access  80040204
-2147220940  lgxError_No_Match_Found  80040234
-2147220938  lgxError_No_Offline_DataFiles  80040236
-2147220947  lgxError_No_Processor  8004022D
-2147220990  lgxError_Not_Deleteable  80040202
-2147220937  lgxError_Not.Done_Uploader  80040237
-2147220986  lgxError_Not_Offline  80040206
-2147220974  lgxError_Not_Program_Mode  80040212
-2147220989  lgxError_Not_Remote_Run  80040203
-2147220959  lgxError_Old_File_Format  80040221
-2147220927  lgxError_Online_Proc_Name_Invalid  80040241
-2147220933  lgxError_Open_Doc_Failed  8004023B
-2147220980  lgxError_Passwords_Dont_Match  8004020C
-2147220944  lgxError_Processor_Faulted  80040230
-2147220985  lgxError_Program_Errors  80040207
-2147220941  lgxError_Pswd_Failed_Login  80040233
-2147220950  lgxError_RackOutOfRange  80040229
-2147220948  lgxError_Rack_SizeOutOfRange  8004022C
-2147220951  lgxError_RAM_Exists  80040229
-2147220943  lgxError_Remote_Emulator  80040231
-2147220930  lgxError_Save_In_Progress  8004023E
-2147220931  lgxError_Secure_Proc_Path_Not_Found  8004023D
-2147220960  lgxError_SLC_Func_Not_Available  80040220
-2147220949  lgxError_SlotOutOfRange  8004022B
-2147220936  lgxError_STX_Not_Supported  80040238
-2147220992  lgxError_Uncalled  80040200
-2147220983  lgxError_Unknown_Proc  80040209
-2147220946  lgxError_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

Appendix

Handling errors
Any of the following lgxErrorTypes may be returned.

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Error Type Definition</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2147220981</td>
<td>lgxError_Abort_Download</td>
<td>8004020B</td>
</tr>
<tr>
<td>-2147220957</td>
<td>lgxError_ACH_Load_Failed</td>
<td>80040223</td>
</tr>
<tr>
<td>-2147220964</td>
<td>lgxError_Autoconfig_Read_Failed</td>
<td>8004021C</td>
</tr>
<tr>
<td>-2147220923</td>
<td>lgxError_Automation_Inhibited</td>
<td>80040245</td>
</tr>
<tr>
<td>-2147220969</td>
<td>lgxError_Backup_Exists</td>
<td>80040217</td>
</tr>
<tr>
<td>-2147220934</td>
<td>lgxError_BOOTP_Cycle_Power</td>
<td>8004023A</td>
</tr>
<tr>
<td>-2147220991</td>
<td>lgxError_Cancel_Delete_All</td>
<td>80040201</td>
</tr>
<tr>
<td>-2147220966</td>
<td>lgxError_Cannot_Connect_To_Proc</td>
<td>8004021A</td>
</tr>
<tr>
<td>-2147220918</td>
<td>lgxError_Cannot_Find_Activation</td>
<td>8004024A</td>
</tr>
<tr>
<td>-2147220978</td>
<td>lgxError_Comm_Problem</td>
<td>8004020E</td>
</tr>
<tr>
<td>-2147220984</td>
<td>lgxError_Comms_Not_Set</td>
<td>80040208</td>
</tr>
<tr>
<td>-2147220932</td>
<td>lgxError_DataTable_Upload_Failed</td>
<td>8004023C</td>
</tr>
<tr>
<td>-2147220961</td>
<td>lgxDB_Create_Failed</td>
<td>8004021F</td>
</tr>
<tr>
<td>-2147220971</td>
<td>lgxError_DB_Error</td>
<td>80040215</td>
</tr>
<tr>
<td>-2147220942</td>
<td>lgxError_Default_Program</td>
<td>80040232</td>
</tr>
<tr>
<td>-2147220977</td>
<td>lgxError.Download_Failed</td>
<td>8004020F</td>
</tr>
<tr>
<td>-2147220975</td>
<td>lgxError_Enable_Forces_Failed</td>
<td>80040211</td>
</tr>
<tr>
<td>-2147220970</td>
<td>lgxError_Failed_Create_DB</td>
<td>80040216</td>
</tr>
<tr>
<td>-2147220973</td>
<td>lgxError_Failed_Del_Temp_DB</td>
<td>80040213</td>
</tr>
<tr>
<td>-2147220972</td>
<td>lgxError_Failed_Init_DB</td>
<td>80040214</td>
</tr>
<tr>
<td>-2147220968</td>
<td>lgxError_Failed_Online</td>
<td>80040218</td>
</tr>
<tr>
<td>-2147220924</td>
<td>lgxError_File_Is_Read_Only</td>
<td>80040244</td>
</tr>
<tr>
<td>-2147220976</td>
<td>lgxError_Forces_Exist</td>
<td>80040210</td>
</tr>
<tr>
<td>-2147220945</td>
<td>lgxError_General_IO_Error</td>
<td>8004022F</td>
</tr>
<tr>
<td>-2147220979</td>
<td>lgxError_Incompatible_Download_Types</td>
<td>8004020D</td>
</tr>
<tr>
<td>-2147220935</td>
<td>lgxError_Incorrect_Class_Priv</td>
<td>80040239</td>
</tr>
<tr>
<td>-2147220920</td>
<td>lgxError_Invalid_Argument</td>
<td>80040248</td>
</tr>
<tr>
<td>-2147220917</td>
<td>lgxError_Invalid_Class</td>
<td>8004024B</td>
</tr>
<tr>
<td>-2147220922</td>
<td>lgxError_Invalid_Data_Type</td>
<td>80040246</td>
</tr>
<tr>
<td>-2147220926</td>
<td>lgxError_Invalid_During_Compare</td>
<td>80040242</td>
</tr>
<tr>
<td>-2147220987</td>
<td>lgxError_Invalid_File_Extension</td>
<td>80040205</td>
</tr>
<tr>
<td>-2147220919</td>
<td>lgxError_Invalid_File_Size</td>
<td>80040249</td>
</tr>
<tr>
<td>-2147220967</td>
<td>lgxError_Invalid_IO_Object</td>
<td>80040219</td>
</tr>
<tr>
<td>-2147220925</td>
<td>lgxError_Invalid_Path_Specified</td>
<td>80040243</td>
</tr>
<tr>
<td>-2147220963</td>
<td>lgxError_Invalid_Rack</td>
<td>8004021D</td>
</tr>
<tr>
<td>-2147220962</td>
<td>lgxError_Invalid_Rack_Config</td>
<td>8004021E</td>
</tr>
<tr>
<td>-2147220958</td>
<td>lgxError_Invalid_RSS_File</td>
<td>80040222</td>
</tr>
<tr>
<td>-2147220921</td>
<td>lgxError_Invalid_Secure_Proc_Operation</td>
<td>80040247</td>
</tr>
<tr>
<td>-2147220955</td>
<td>lgxError_Invalid_SLC_File</td>
<td>80040225</td>
</tr>
<tr>
<td>-2147220952</td>
<td>lgxError_Library_Failure</td>
<td>80040228</td>
</tr>
<tr>
<td>-2147220956</td>
<td>lgxError_Library_Load_Failed</td>
<td>80040224</td>
</tr>
<tr>
<td>-2147220953</td>
<td>lgxError_Library_Partial_Load_Failed</td>
<td>80040227</td>
</tr>
<tr>
<td>-2147220954</td>
<td>lgxError_Library_Warnings_Exist</td>
<td>80040226</td>
</tr>
<tr>
<td>-2147220939</td>
<td>lgxError_Multiple_Files_Found</td>
<td>80040235</td>
</tr>
<tr>
<td>-2147220929</td>
<td>lgxError_Must_Be_Offline</td>
<td>8004023F</td>
</tr>
<tr>
<td>-2147220965</td>
<td>lgxError_No_AutoConfig</td>
<td>8004021B</td>
</tr>
<tr>
<td>-2147220928</td>
<td>lgxError_No_Checksum</td>
<td>80040240</td>
</tr>
<tr>
<td>-2147220982</td>
<td>lgxError_No_Controller_Response</td>
<td>8004020A</td>
</tr>
</tbody>
</table>

continued on next page...
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Description</th>
<th>Error Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-214720988</td>
<td>lgxError_No_Future_Access</td>
<td>80040204</td>
<td></td>
</tr>
<tr>
<td>-214720940</td>
<td>lgxError_No_Match_Found</td>
<td>80040234</td>
<td></td>
</tr>
<tr>
<td>-214720938</td>
<td>lgxError_No_Offline_DataFiles</td>
<td>80040236</td>
<td></td>
</tr>
<tr>
<td>-214720947</td>
<td>lgxError_No_Processor</td>
<td>8004022D</td>
<td></td>
</tr>
<tr>
<td>-214720990</td>
<td>lgxError_Not_Deleteable</td>
<td>80040202</td>
<td></td>
</tr>
<tr>
<td>-214720937</td>
<td>lgxError_Not_Done_Uploading</td>
<td>80040237</td>
<td></td>
</tr>
<tr>
<td>-214720986</td>
<td>lgxError_Not_Offline</td>
<td>80040206</td>
<td></td>
</tr>
<tr>
<td>-214720974</td>
<td>lgxError_Not_Program_Mode</td>
<td>80040212</td>
<td></td>
</tr>
<tr>
<td>-214720989</td>
<td>lgxError_Not_Remote_Run</td>
<td>80040203</td>
<td></td>
</tr>
<tr>
<td>-214720959</td>
<td>lgxError_Old_File_Format</td>
<td>80040221</td>
<td></td>
</tr>
<tr>
<td>-214720927</td>
<td>lgxError_Online_Proc_Name_Invalid</td>
<td>80040241</td>
<td></td>
</tr>
<tr>
<td>-214720933</td>
<td>lgxError_Open_Doc_Failed</td>
<td>8004023B</td>
<td></td>
</tr>
<tr>
<td>-214720980</td>
<td>lgxError_Passwords_Dont_Match</td>
<td>8004020C</td>
<td></td>
</tr>
<tr>
<td>-214720944</td>
<td>lgxError_Processor_Faulted</td>
<td>80040230</td>
<td></td>
</tr>
<tr>
<td>-214720985</td>
<td>lgxError_Program_Errors</td>
<td>80040207</td>
<td></td>
</tr>
<tr>
<td>-214720941</td>
<td>lgxError_Pswd_Failed_Login</td>
<td>80040233</td>
<td></td>
</tr>
<tr>
<td>-214720950</td>
<td>lgxError_Rack_Out_Of_Range</td>
<td>8004022A</td>
<td></td>
</tr>
<tr>
<td>-214720948</td>
<td>lgxError_Rack_Size_Out_Of_Range</td>
<td>8004022C</td>
<td></td>
</tr>
<tr>
<td>-214720951</td>
<td>lgxError_RAM_Edits_Exist</td>
<td>80040229</td>
<td></td>
</tr>
<tr>
<td>-214720943</td>
<td>lgxError_Remote_Emulator</td>
<td>80040231</td>
<td></td>
</tr>
<tr>
<td>-214720930</td>
<td>lgxError_Save_In_Progress</td>
<td>8004023E</td>
<td></td>
</tr>
<tr>
<td>-214720931</td>
<td>lgxError_Secure_Proc_Path_Not_Found</td>
<td>8004023D</td>
<td></td>
</tr>
<tr>
<td>-214720960</td>
<td>lgxError_SLC_Func_Not_Available</td>
<td>80040220</td>
<td></td>
</tr>
<tr>
<td>-214720949</td>
<td>lgxError_SLOT_Out_Of_Range</td>
<td>8004022B</td>
<td></td>
</tr>
<tr>
<td>-214720936</td>
<td>lgxError_STX_Not_Supported</td>
<td>80040238</td>
<td></td>
</tr>
<tr>
<td>-214720992</td>
<td>lgxError_Unexpected</td>
<td>80040200</td>
<td></td>
</tr>
<tr>
<td>-214720983</td>
<td>lgxError_Unknown_Proc</td>
<td>80040209</td>
<td></td>
</tr>
<tr>
<td>-214720946</td>
<td>lgxError_Unsupported_Feature</td>
<td>8004022E</td>
<td></td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Properties</th>
<th>RSLogix 5</th>
<th>RSLogix 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumberOfElements</td>
<td>Read/Write. Returns the number of elements in the data file.</td>
<td>Read Only.</td>
</tr>
<tr>
<td>Reserved</td>
<td>Does not exist.</td>
<td>Read Only. Returns True if the data file is reserved.</td>
</tr>
<tr>
<td>ReadPrivilege</td>
<td>Read Only. Returns whether the program file is read-enabled under the current privilege class.</td>
<td>Does not exist.</td>
</tr>
<tr>
<td>WritePrivilege</td>
<td>Read Only. Returns whether the program file is write-enabled under the current privilege class.</td>
<td>Does not exist.</td>
</tr>
</tbody>
</table>

**ProgramFile object**

<table>
<thead>
<tr>
<th>Properties</th>
<th>RSLogix 5</th>
<th>RSLogix 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProtectionSupported</td>
<td>Does not exist.</td>
<td>Read only. Returns whether or not protection is supported by this program file.</td>
</tr>
<tr>
<td>Reserved</td>
<td>Does not exist.</td>
<td>Read Only. Returns True if the program file is reserved.</td>
</tr>
<tr>
<td>ReadPrivilege</td>
<td>Read Only. Returns whether the program file is read-enabled under the current privilege class.</td>
<td>Does not exist.</td>
</tr>
<tr>
<td>WritePrivilege</td>
<td>Read Only. Returns whether the program file is write-enabled under the current privilege class.</td>
<td>Does not exist.</td>
</tr>
</tbody>
</table>

**ReportOptions object**

<table>
<thead>
<tr>
<th>Properties</th>
<th>RSLogix 5</th>
<th>RSLogix 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multipoint</td>
<td>Does not exist.</td>
<td>Read/Write. If set to True, a multipoint monitor report is included.</td>
</tr>
</tbody>
</table>

**LogixProject object**

<table>
<thead>
<tr>
<th>Properties</th>
<th>RSLogix 5</th>
<th>RSLogix 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>PasswordPrivilegeCfg</td>
<td>Returns the Password/Privilege configuration for the RSLogix 5 processor.</td>
<td>Does not exist.</td>
</tr>
</tbody>
</table>
### Processor object

<table>
<thead>
<tr>
<th>Properties</th>
<th>RSLogix 5</th>
<th>RSLogix 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentPLC5MemSize (long)</td>
<td>Read only. Returns the processor memory size.</td>
<td>Does not exist.</td>
</tr>
<tr>
<td>HasPasswordPrivileges (boolean)</td>
<td>Read only. Returns if the processor supports privileges.</td>
<td>Does not exist.</td>
</tr>
<tr>
<td>NumberOfMemSizeChoices (short)</td>
<td>Read only. Returns how many choices of memory size you have for the current processor.</td>
<td>Does not exist.</td>
</tr>
<tr>
<td>ProgramID (integer)</td>
<td>Does not exist.</td>
<td>Read Only. Returns the 4-byte error check (CRC) of the program.</td>
</tr>
<tr>
<td>Series (integer)</td>
<td>Read/Write. Sets or returns the series # of the processor.</td>
<td>Does not exist.</td>
</tr>
<tr>
<td>Revision (integer)</td>
<td>Read/Write. Sets or returns the revision # of the processor.</td>
<td>Does not exist.</td>
</tr>
<tr>
<td>SubRevision (integer)</td>
<td>Read/Write. Sets or returns the subrevision # of the processor.</td>
<td>Does not exist.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th></th>
<th>RSLogix 5</th>
<th>RSLogix 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetPLC5MemSizeChoiceByIndex (long)</td>
<td>Gets the size of the processor's memory.</td>
<td>Does not exist.</td>
</tr>
<tr>
<td>SetPLC5MemSize</td>
<td>Sets the memory size of the processor.</td>
<td>Does not exist.</td>
</tr>
</tbody>
</table>

### Ladder object

<table>
<thead>
<tr>
<th>Properties</th>
<th>RSLogix 5</th>
<th>RSLogix 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProtectionSupported (Boolean)</td>
<td>Does not exist.</td>
<td>Read Only. Returns the attribute of protection supported by this ladder file.</td>
</tr>
<tr>
<td>ReadPrivilege (Boolean)</td>
<td>Read Only. Returns whether the ladder file is read-enabled under the current privilege class.</td>
<td>Does not exist.</td>
</tr>
<tr>
<td>WritePrivilege (Boolean)</td>
<td>Read Only. Returns whether the ladder file is write-enabled under the current privilege class.</td>
<td>Does not exist.</td>
</tr>
</tbody>
</table>
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

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
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
element of use • 51
methods • 49
properties • 46
Processor property • 27
ProcessorInfo property • 131
ProcessorMode property • 48
ProgFile property • 166
ProgramFile object • 65
about • 65
element of use • 68
properties • 66
ProgramFileList property • 131
ProgramFileRange property • 132
ProgramFiles Collection • 57
ProgramFiles collection
about • 57
element 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
element 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
SetClassPassword method • 148
SetDataValue method • 185
SetDefaultClass method • 78
SetDefaultClass method • 187
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
  lgxUpDownloadAction • 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