# Contents

## Preface ........................................................................................................ 21
- Documentation Conventions ................................................................. 22
- Related Publications .............................................................................. 23
- Customer Support .................................................................................. 23
- Information You Should Have .............................................................. 24
- User Feedback ........................................................................................ 25
- Information Builders Consulting and Training ....................................... 25

## 1. Setting Up Your WebFOCUS Maintain Project ..................................... 27
- Opening WebFOCUS Maintain ............................................................... 27
  - Procedure: How to Open the Maintain Development Environment. .... 28
- Applications and Projects ..................................................................... 28
  - Procedure: How to Create a Project. .................................................. 28
- What Are the Building Blocks of a WebFOCUS Maintain Project? .......... 29
  - Creating New Project Files. ............................................................... 29
    - Procedure: How to Create a Project Component. ......................... 29
  - Adding Existing Files to Your Project. ............................................ 30
    - Procedure: How to Add an Existing File to Your Project. .......... 30
    - Procedure: How to Add an Existing File From Another WebFOCUS Server to Your Project. ................................. 31
- What Other Types of Files Can Go Into a WebFOCUS Maintain Project? 32
- Defining the Flow of a Project .............................................................. 32
- Editing Projects in the Maintain Development Environment ................ 33
  - Procedure: How to Save Your Project or a Component in Your Project. 34
  - Procedure: How to Save Your Project Automatically. .................... 34
  - Procedure: How to Change the Project Path. .................................... 34
  - Procedure: How to View New File Types in the Explorer. ............... 35
  - Procedure: How to View or Hide All Files in the Project Path. .......... 35
  - Procedure: How to Rename a Project or Project Component. ........... 35
  - Procedure: How to Delete a Project or Project Component. ............... 36
  - Procedure: How to Remove a Project Component From a Project. .... 36
- Viewing System Messages in the Output Window .................................. 36
  - Procedure: How to Open and Close the Output Window. ................. 37
- Searching for Text in Your Project ......................................................... 37
Procedure: How to Search for Text in Your Project ........................................... 37
Editing Project Components as Text ............................................................... 38
Procedure: How to Open a Project Component as Text ............................... 39
Customizing Parsing Activity ......................................................................... 39
Procedure: How to Customize Parsing Activity .......................................... 39
Common Editing Functions ............................................................................ 39
Procedure: How to Enter Text ....................................................................... 39
Procedure: How to Select a Range of Text ..................................................... 40
Procedure: How to Select All Text .................................................................. 40
Procedure: How to Move Text to Another Location ....................................... 40
Procedure: How to Copy Text to Another Location ........................................ 40
Procedure: How to Undo or Redo an Action .................................................. 40
Procedure: How to Find Text ......................................................................... 41
Procedure: How to Replace Text .................................................................... 41
Procedure: How to Preview an Application Component as Text .................. 42
Procedure: How to Print an Application Component as Text ....................... 42
Using Bookmarks to Move Within the Editor .................................................. 42
Procedure: How to Apply or Remove a Bookmark ......................................... 42
Procedure: How to Jump to a Bookmark ......................................................... 43
Procedure: How to Apply Multiple Bookmarks .............................................. 43
Procedure: How to Remove All Bookmarks .................................................. 43
Incorporating Data Sources Into Your Project .............................................. 43
Procedure: How to Edit a Data Source Description From the MDE .............. 44
Procedure: How to Edit a Data Source Description From Developer Studio .... 44
Procedure: How to Supply a WebFOCUS DBA Password ............................... 45
Procedure: How to View the Structure of a Data Source ............................... 45
Using the Resource Wizard ............................................................................ 46
Reference: Resource Wizard: Specify Path for Image ..................................... 47
Reference: Resource Wizard: Specify Path for HTML File ............................. 48
Reference: Resource Wizard: Specify Path for Java Applet ........................... 49
Reference: Resource Wizard: Define Parameters for Java Applet ................. 50
Reference: Resource Wizard: Specify ActiveX Controls ................................ 51
Reference: Open Dialog Box ......................................................................... 52
Reference: Resource Wizard: Specify a Unique Name .................................... 53
Reference: Copy Resource Dialog Box ............................................................ 54
2. Developing Procedures

Overview

Procedure: How to Designate the Starting Procedure.
Procedure: How to Edit the Source Code of a Procedure.
Procedure: How to Use the Language Wizard.
Procedure: How to View the Description of a Procedure Component.
Procedure: How to View Your Procedure Components in Folders.

Specifying Data Sources for Your Procedure

Procedure: How to Specify Data Sources for Your Procedure.
Reference: Use These Data Sources in Procedure Dialog Box.

Using Functions in Procedures

Procedure: How to Create a Function.
Procedure: How to Edit the Name, Arguments, Return Value, and Description of a Function.
Procedure: How to Edit a Function.
Procedure: How to Move a Function to Another Procedure.
Procedure: How to Copy a Function to Another Procedure.
Reference: Function Editor: Signature Tab.
Reference: Function Editor: Description Tab.

Using Variables in Procedures

Procedure: How to Create a Variable in a Procedure.
Procedure: How to Edit a Variable.
Procedure: How to Edit Source Codes in a Variable.
Procedure: How to Move a Variable to Another Procedure.
Procedure: How to Copy a Variable to Another Procedure.
Reference: Variable Editor: Declaration Tab.
Reference: Variable Editor: x=? Initialize Tab.
Reference: Variable Editor: Description Tab.

Using Data Source Stacks in Procedures

Procedure: How to Create a Data Source Stack Explicitly Using the Stack Editor.
Procedure: How to Edit Source Codes in a Data Source Stack.
Reference: Stack Editor: Definition Tab.
Contents

Reference: Stack Editor: Description Tab. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
Reference: Create New Stack Dialog Box. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
Using the Type Wizard . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
Procedure: How to Apply Alphanumeric Format. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
Procedure: How to Apply Date Format. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
Procedure: How to Apply Numeric Format. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
Procedure: How to Apply a User-defined Class. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81
Using Import Modules . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81
Procedure: How to Use an Import Module in a Procedure. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 82
Reference: Import These Modules Into Procedure Dialog Box. . . . . . . . . . . . . . . . . . . . . . . . . 83
Testing Procedures With the Run Procedure Option . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
Procedure: How to Test Procedures With the Run Procedure Option. . . . . . . . . . . . . . . . . . . . 84

3. Using the Form Editor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85
Form Editor Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
Layout of the Form Editor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
Using the Controls Palette . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
Procedure: How to Place a Control on Your Form. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
Using the Property Sheet . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 89
Procedure: How to Change a Property. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91
Procedure: How to Set Properties Dynamically. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91
Using Drawing Aids . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 92
Procedure: How to Display a Drawing Grid and Guidelines. . . . . . . . . . . . . . . . . . . . . . . . . . . . 92
Procedure: How to Change Drawing Grid Settings. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 92
Procedure: How to Display Rulers. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
Reference: Grid Settings Dialog Box. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
Selecting Multiple Controls . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94
Procedure: How to Select Multiple Controls. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94
Cutting, Copying, Pasting, and Duplicating Controls . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94
Procedure: How to Cut a Control. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 95
Procedure: How to Copy a Control. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 95
Procedure: How to Paste a Control. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96
Procedure: How to Duplicate a Control. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96
Resizing Controls . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 97
Procedure: How to Make Controls the Same Width. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 97

6

WebFOCUS


<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procedure: How to Create a Form.</td>
<td>110</td>
</tr>
<tr>
<td>Procedure: How to Edit a Form.</td>
<td>111</td>
</tr>
<tr>
<td>Procedure: How to Rename a Form.</td>
<td>111</td>
</tr>
<tr>
<td>Exporting and Importing Forms.</td>
<td>111</td>
</tr>
<tr>
<td>Procedure: How to Export a Form.</td>
<td>112</td>
</tr>
<tr>
<td>Procedure: How to Import a Form.</td>
<td>112</td>
</tr>
<tr>
<td>Copying a Form Within a Procedure</td>
<td>112</td>
</tr>
<tr>
<td>Procedure: How to Copy a Form Within a Procedure.</td>
<td>112</td>
</tr>
<tr>
<td>Changing Form Properties</td>
<td>112</td>
</tr>
<tr>
<td>Using Cascading Style Sheets</td>
<td>113</td>
</tr>
<tr>
<td>Procedure: How to Set a Style Sheet Using the Properties Sheet.</td>
<td>113</td>
</tr>
<tr>
<td>Syntax: How to Apply a Style Sheet to Your Form.</td>
<td>114</td>
</tr>
<tr>
<td>Syntax: How to Apply a Class to a Control on Your Form.</td>
<td>114</td>
</tr>
<tr>
<td>Dynamically Manipulating Forms at Run Time</td>
<td>114</td>
</tr>
<tr>
<td>Displaying Forms at Run Time</td>
<td>114</td>
</tr>
<tr>
<td>Syntax: How to Make a Form Active.</td>
<td>115</td>
</tr>
<tr>
<td>Syntax: How to Display a Non-Persistent Form.</td>
<td>116</td>
</tr>
<tr>
<td>Syntax: How to Display a Form Without Making it Active.</td>
<td>116</td>
</tr>
<tr>
<td>The Active Form.</td>
<td>116</td>
</tr>
<tr>
<td>The Non-Persistent Form</td>
<td>116</td>
</tr>
<tr>
<td>Closing Forms at Run Time</td>
<td>117</td>
</tr>
<tr>
<td>Syntax: How to Close a Form.</td>
<td>118</td>
</tr>
<tr>
<td>Syntax: How to Close All Forms.</td>
<td>119</td>
</tr>
<tr>
<td>Manipulating Form Properties</td>
<td>119</td>
</tr>
<tr>
<td>Syntax: How to Determine the Value of a Form Property.</td>
<td>120</td>
</tr>
<tr>
<td>Syntax: How to Set the Value of a Form Property.</td>
<td>120</td>
</tr>
<tr>
<td>Syntax: How to Reset a Form.</td>
<td>121</td>
</tr>
<tr>
<td>Exiting Your Application at Run Time.</td>
<td>121</td>
</tr>
<tr>
<td>Syntax: How to Close the Application.</td>
<td>121</td>
</tr>
<tr>
<td>How WebFOCUS Maintain Saves Your Forms</td>
<td>122</td>
</tr>
<tr>
<td>Using a Driver Procedure</td>
<td>122</td>
</tr>
</tbody>
</table>

**5. Defining Events and Event Handlers** ............................................. 125

Using the Event Handler Editor ................................................................ 125
Procedure: How to Open the Event Handler Editor.                          | 126
<table>
<thead>
<tr>
<th>Reference: Event Handler Editor</th>
<th>127</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events</td>
<td>129</td>
</tr>
<tr>
<td>Common Combinations of Events and Controls</td>
<td>131</td>
</tr>
<tr>
<td>Using a Maintain Function as an Event Handler</td>
<td>131</td>
</tr>
<tr>
<td>Procedure: How to Execute a Maintain Function From the Event Handler Editor</td>
<td>132</td>
</tr>
<tr>
<td>Using Script Functions as Event Handlers</td>
<td>132</td>
</tr>
<tr>
<td>About JavaScript and VBScript.</td>
<td>133</td>
</tr>
<tr>
<td>Using Script Functions in Your Project</td>
<td>133</td>
</tr>
<tr>
<td>Procedure: How to Write a JavaScript or VBScript Function in the Event Handler Editor</td>
<td>134</td>
</tr>
<tr>
<td>Procedure: How to Assign a JavaScript or VBScript Function in a Script Library to an Event</td>
<td>134</td>
</tr>
<tr>
<td>Using Script Functions for Validation</td>
<td>135</td>
</tr>
<tr>
<td>Syntax: How to Use the JavaScript Alert Function</td>
<td>136</td>
</tr>
<tr>
<td>Syntax: How to Use the JavaScript Confirm Function</td>
<td>136</td>
</tr>
<tr>
<td>Syntax: How to Use the JavaScript Prompt Function</td>
<td>137</td>
</tr>
<tr>
<td>Syntax: How to Use the IWCTrigger Function to Call a Maintain Function From Your Script Handler</td>
<td>137</td>
</tr>
<tr>
<td>Using Script Functions With HTML Tables</td>
<td>141</td>
</tr>
<tr>
<td>Syntax: How to Determine Row Number, Column Number, or Cell Value When an HTML Table Is Clicked Using a Script</td>
<td>141</td>
</tr>
<tr>
<td>Using Script Functions to Turn Secure Sockets Layer On and Off</td>
<td>142</td>
</tr>
<tr>
<td>Using Script Functions to Make Controls Visible</td>
<td>142</td>
</tr>
<tr>
<td>Syntax: How to Make a Control Visible</td>
<td>142</td>
</tr>
<tr>
<td>Managing Script Libraries.</td>
<td>142</td>
</tr>
<tr>
<td>Procedure: How to Edit a Script Library</td>
<td>143</td>
</tr>
<tr>
<td>Procedure: How to Associate a Script Library With a Form</td>
<td>143</td>
</tr>
<tr>
<td>Running and Debugging JavaScript and VBScript Functions</td>
<td>143</td>
</tr>
<tr>
<td>Using Web Links as Event Handlers</td>
<td>143</td>
</tr>
<tr>
<td>Procedure: How to Create a Web Link</td>
<td>144</td>
</tr>
<tr>
<td>Procedure: How to Use the URL Wizard to Define an HTTP Web Link</td>
<td>147</td>
</tr>
<tr>
<td>Procedure: How to Use the URL Wizard to Define an FTP Web Link</td>
<td>147</td>
</tr>
<tr>
<td>Procedure: How to Use the URL Wizard to Define a WebFOCUS Report as a Web Link</td>
<td>148</td>
</tr>
<tr>
<td>Procedure: How to Use the URL Wizard to Define an Email Web Link</td>
<td>148</td>
</tr>
<tr>
<td>Procedure: How to Add an HTML File to Your Project as a Resource</td>
<td>149</td>
</tr>
</tbody>
</table>
6. Developing and Using Controls ................................................................. 163

Which Control Should You Use? ................................................................. 164
Using Buttons ............................................................................................... 165
Procedure: How to Place a Button on Your Form ......................................... 165
Changing Button Properties ........................................................................ 165
Using Check Boxes ....................................................................................... 167
Procedure: How to Place a Check Box on Your Form ..................................... 167
Syntax: How to Set the Value of a Check Box Dynamically ............................ 168
Procedure: How to Trigger an Action When an End User Selects a Check Box. 168
Reference: Set Check Box State Dialog Box .................................................. 169
Changing Check Box Properties .................................................................... 170
Using Combo Boxes and List Boxes ............................................................. 172
Procedure: How to Place a Combo Box or a List Box on Your Form ............... 172
Procedure: How to Enter Items Manually in the List Source Dialog Box .......... 173
Syntax: How to Set the Value of a Combo Box or List Box Dynamically ......... 174
Syntax: How to Preselect Items in a Multi-Select List Box by Value ............... 174
Procedure: How to Trigger an Action When Selecting From a Combo Box or List Box. 175
Reference: List Source Dialog Box ............................................................... 176
Reference: Enter a List Item Dialog Box ...................................................... 179
Reference: Binding the Selection Result Dialog Box ..................................... 179
Changing Combo Box or List Box Properties ......................................................... 181

Using Edit Boxes and Multi-Edit Boxes ................................................................. 182
Procedure: How to Place an Edit Box or Multi-Edit Box on Your Form ............... 182
Reference: Insert Text Dialog Box ........................................................................... 184
Changing Edit or Multi-Edit Box Properties ......................................................... 185
Using Prompted Edit Boxes ..................................................................................... 187
  Reference: Select Stack Columns Dialog Box ....................................................... 188
  Reference: Select Segment Fields Dialog Box ..................................................... 189

Using Frames ........................................................................................................... 191
Procedure: How to Launch a Web Link in a Frame ................................................. 191
Reference: URL Link Dialog Box ............................................................................ 192
Changing Frame Properties ..................................................................................... 192

Using Grids and HTML Tables ............................................................................... 193
Procedure: How to Place a Read/Write Grid or an HTML Table on a Form ......... 194
Reference: Control Columns Dialog Box ................................................................. 195
Reference: Table Column/Grid Column Properties Dialog Box ......................... 197
Reference: Events Available in a Read/Write Grid ............................................... 199
Changing the Properties of a Grid or HTML Table ............................................. 201
Creating Links in HTML Tables .............................................................................. 204
  Procedure: How to Create Links in an HTML Table Column ......................... 204
  Syntax: How to Determine Row Number, Column Number, or Value of the Cell When an HTML Table Is Clicked ...................................................... 205
  Syntax: How to Reset FocIndex in a Data Source Stack for an HTML Table ....... 205
Using Functions in the Read/Write Grids ................................................................. 205
  Reference: Property Functions in the ActiveX Read/Write Grid ......... .............. 206
  Syntax: How to Use the Mask Function With the ActiveX Grid ....................... 207
  Reference: Property Functions in the Read/Write JS Grid ................................. 212

Using Group Boxes ................................................................................................ 217
Procedure: How to Add a Group Box Control to Your Form .............................. 217
Changing Group Box Properties ............................................................................. 217

Using HTML Objects .............................................................................................. 218
Procedure: How to Place an HTML Object on Your Form .................................. 219
Reference: HTML Content Source Dialog Box .................................................... 220
Changing HTML Object Properties ................................................................ .... 222

Using Images .......................................................................................................... 223
Contents

Procedure: How to Add an Image to Your Form ........................................... 223
Procedure: How to Add an Image to Your Project as a Resource .................. 224
Reference: Image Source Dialog Box ......................................................... 225
Tips for Using Images in Your Applications .............................................. 227
Changing Image Properties ........................................................................ 228
Using Image Maps ....................................................................................... 229
  Procedure: How to Define an Image Map ................................................ 230
  Reference: Image Map Dialog Box .......................................................... 231
Changing Images at Run Time Using ImageDown and ImageOver ............. 232
  Procedure: How to Change an Image When End Users Click It ............... 233
Using Java Applets ..................................................................................... 233
  Procedure: How to Place a Java Applet on Your Form ............................ 233
  Procedure: How to Add a Java Applet to Your Project as a Resource ....... 234
Syntax: How to Change Java Applet Parameters Dynamically at Run Time ... 235
  Reference: Parameters Dialog Box ......................................................... 236
  Reference: Enter a Parameter Dialog Box ............................................. 237
Changing Java Applet Properties .............................................................. 238
Using Lines .................................................................................................. 239
  Procedure: How to Place a Line on Your Form ....................................... 239
  Changing Line Properties .................................................................. 239
Using Menus ................................................................................................. 240
  Procedure: How to Place a Menu Bar on Your Form ............................. 240
  Procedure: How to Create Pull-Down Menus and Submenus ................. 241
  Reference: The Menu Items Dialog Box ............................................... 244
  Changing Menu Bar Properties ............................................................ 245
Using Radio Buttons .................................................................................. 246
  Procedure: How to Place a Radio Button on Your Form ......................... 247
  Syntax: How to Set the Value of a Radio Button Group Dynamically ....... 248
  Procedure: How to Trigger an Action When an End User Selects an Item in a Radio Button Group ..................................................... 248
  Changing Radio Button Properties ..................................................... 249
  Determining the Layout of Your Radio Buttons ..................................... 250
Using Text .................................................................................................... 251
  Procedure: How to Place Text on Your Form .......................................... 252
  Changing Text Properties .................................................................... 252
Using Text Controls as Hyperlinks. ................................................................. 253
  Procedure: How to Turn Your Text Control Into a Hyperlink. ....................... 253
Using ActiveX Controls ......................................................................................... 253
  Procedure: How to Place an ActiveX Control on Your Form. ............................ 254
  Procedure: How to Add an ActiveX Control to Your Project as a Resource. ....... 254
  Syntax: How to Pass the Value of an ActiveX Control Property to a Maintain Function. 255
Reference: Insert ActiveX Control Dialog Box. ...................................................... 256
Reference: Insert ActiveX Control Resource. ......................................................... 257
Changing ActiveX Control Properties. ................................................................. 257
Unstable ActiveX Controls. ...................................................................................... 258
Dynamically Manipulating Controls at Run Time .................................................. 259
  Syntax: How to Determine the Value of a Control Property. ............................... 259
  Syntax: How to Set the Value of a Control Property. ........................................... 259
Defining Colors for Your Form and Controls ....................................................... 260
  Procedure: How to Add Colors. ...................................................................... 261
  Procedure: How to Define Custom Colors. ....................................................... 261
  Syntax: How to Set the Value of a Control Color Property. ............................... 262
Assigning Help to Your Forms and Controls ......................................................... 263
  Procedure: How to Assign Help to Your Forms and Controls. ......................... 263
  Reference: Help Dialog Box. ........................................................................ 263
Assigning Tab Order to Controls .......................................................................... 264
  Procedure: How to Assign Tab Order to Controls. ............................................ 264
  Reference: Tab Order Dialog Box. ................................................................. 265

7. Form and Control Properties Reference .......................................................... 267
   (GroupCode) Property ....................................................................................... 267
   (Name) Property .............................................................................................. 267
   Alignment Property .......................................................................................... 267
   Alt Property ..................................................................................................... 268
   AlternateRowColor Property .......................................................................... 268
   BackColor Property ......................................................................................... 269
   BackColorOver Property .................................................................................. 270
   BackgroundImage Property ............................................................................. 270
   BodyRowHeight Property ............................................................................... 270
   Border Property .............................................................................................. 271
8. Executing Other Procedures ........................................ 297
   The Advantages of Modularizing Source Code ........................................ 297
   Using the CALL and EXEC Commands ........................................ 298
      Syntax: How to Use the CALL and EXEC Commands. ........................................ 298
      Optional PATH Keyword for CALL and EXEC Statements. ........................................ 299
      Procedure: How to Introduce a PATH Keyword Into a CALL or EXEC Statement. .... 299
   Executing a Procedure on Another Server: AT Server ........................................ 300
   Keeping or Terminating the Server Session: KEEP/DROP ........................................ 301
   Passing Parameters Between Maintain Procedures: FROM...INTO ........................................ 301
      Specifying FROM and INTO Parameters in the Calling Procedure. ........................................ 302
      Specifying FROM and INTO Parameters in the Called Procedure. ........................................ 302
      Syntax: How to Call a Maintain Procedure With FROM and INTO Parameters. ........................................ 302
      Defining FROM and INTO Parameters in Calling and Called Maintain Procedures. .................. 303
Specifying FROM and INTO Parameters for a Called Procedure Using the Procedure Parameters Dialog Box. .................................................. 304
Procedure: How to Specify Parameters for a Called Procedure Using the Procedure Parameters Dialog Box. .................................................. 305
Using the Language Wizard to Specify Parameters When Calling a Maintain Procedure. .................................................. 305
Procedure: How to Use the Language Wizard to Specify Parameters When Calling a Maintain Procedure. .................................................. 305
Accessing Data Sources in the Called Procedure. .......................... 311
Data Source Position in Child Procedures. .................................. 311
Passing Parameters Between Maintain and WebFOCUS Procedures: FROM...INTO .................. 312
Reference: Usage Notes for Input and Output Parameters in the EXEC Command .................. 312
Optimizing Performance: Data Continuity and Memory Management .................................................. 313

What Can WebFOCUS Procedures Do? .................................. 315
Incorporating WebFOCUS Procedures Into Your Project .................. 316
Procedure: How to Edit the Source Code of a WebFOCUS Procedure .................................................. 316
Executing WebFOCUS Procedures From Maintain Procedures .................. 316
Using WebFOCUS Report Output in Maintain .......................... 317
Procedure: How to Display the Output From a FOCUS Report on a Form .................................................. 317
Procedure: How to Use the Output From a WebFOCUS Report .................................................. 318
Executing Maintain Procedures From WebFOCUS Report Procedures .................. 322
Procedure: How to Execute Maintain Procedures From WebFOCUS Report Procedures .................................................. 322
Using Variable Binding From WebFOCUS Reports .................................................. 323
Procedure: How to Use the Variable Binding Feature .................................................. 323
Reference: Variable Binding Limitations .................................. 323
Reference: Importing WebFOCUS Parameters Into Maintain Dialog Box. .................................................. 324

10. Developing Classes and Objects ........................................ 327
What Are Classes and Objects? .................................. 327
Class Properties: Member Variables and Member Functions .................................................. 328
Inheritance: Superclasses and Subclasses .................................. 330
Defining Classes .................................................. 330
Procedure: How to Define a Class Using the Class Editor .................................................. 331
Procedure: How to Define a Subclass Using the Class Editor .................................................. 332
Syntax: How to Define a Class or Subclass Using the DESCRIBE Command .................................................. 333
| Procedure: How to Edit a Class Definition. | ................................................................. 334 |
| Procedure: How to Edit a Class’s Source Code. | ............................................................. 334 |
| Procedure: How to Rename a Class, Member Variable, or Member Function. | ................................................. 335 |
| Procedure: How to Delete a Class, Member Variable, or Member Function. | ................................................... 335 |
| Reference: New Class and Edit Class Dialog Boxes: General Tab. | ....................................................... 336 |
| Reference: New Class and Edit Class Dialog Boxes: Functions Tab. | ..................................................... 337 |
| Reference: New Class and Edit Class Dialog Boxes: Variables Tab. | ..................................................... 338 |
| Reference: New Class and Edit Class Dialog Boxes: Description Tab. | ..................................................... 340 |
| Reusing Classes: Class Libraries | ................................................................. 340 |
| Syntax: How to Import a Class Library Using the MODULE IMPORT Command. | ................................................. 341 |
| Declaring Objects | ............................................................... 341 |
| Procedure: How to Declare an Object Using the Variable Editor. | ............................................. 341 |
| Syntax: How to Declare an Object Using the DECLARE Command. | ............................................. 342 |

**11. Running WebFOCUS Maintain Applications** ................................................................. 345

| Compiling WebFOCUS Maintain Procedures | ................................................................. 345 |
| Executing Maintain Procedures From Outside the Maintain Development Environment | ................................................. 346 |
| Syntax: How to Execute a WebFOCUS Maintain Procedure. | ........................................... 346 |
| Using the WFMSTART.HTML Launch Page. | ............................................................. 347 |
| Security and Running WebFOCUS Maintain Applications | ..................................................... 348 |
| Supplying DBA Information at Run Time. | .............................................................. 348 |
| Store and Retrieve Credentials With Web Browser Cookies. | ................................................. 348 |
| Customizing Key Functions and JavaScript for WebFOCUS Maintain Applications | ............................................. 348 |
| Key Restriction | ................................................................. 349 |
| Customized JavaScript Functions | ................................................................. 349 |
| Closing WebFOCUS Maintain Applications | .............................................................. 350 |

**12. Designing Applications for Scalability and Performance** ................................................................. 351

| Application Partitioning | ................................................................. 351 |
| Presentation Logic | ................................................................. 352 |
| Database Logic | ................................................................. 353 |
| Miscellaneous Business Logic | ............................................................. 354 |
| EXEC Command, CALL Command, and System Resources | .............................................. 355 |
| Use When Appropriate | ................................................................. 355 |
| Using the Best Parts | ................................................................. 356 |
| Persistence Management | ................................................................. 356 |
### Development Standards

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>358</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tips for Designing Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>358</td>
</tr>
</tbody>
</table>

### 13. Ensuring Transaction Integrity

<table>
<thead>
<tr>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>359</td>
</tr>
</tbody>
</table>

#### Transaction Integrity Overview

- Procedure: How to Process a Logical Transaction. 360
- Why Is Transaction Integrity Important? 360

#### Defining a Transaction

- When Does a Data Source Command Cause a Transaction to Fail? 362
- Canceling a Transaction. 362
- Transactions and Data Source Position. 362
- How Large Should a Transaction Be? 363
- Designing Transactions That Span Procedures. 363
- Designing Transactions That Span Data Source Types. 364
- Designing Transactions in Multi-Server Applications. 365
- When an Application Ends With an Open Transaction. 365

#### Evaluating Whether a Transaction Was Successful

- Concurrent Transaction Processing 365

#### Ensuring Transaction Integrity for FOCUS Data Sources

- Setting COMMIT. 368
- Syntax: How to Set COMMIT. 369
- Sharing Access to FOCUS Data Sources. 369
- How the FOCUS Database Server and Change-verification Work. 370
- Selecting Which Segments Will Be Verified for Changes. 371
- Syntax: How to Set PATHCHECK. 371
- Identifying the FOCUS Database Server. 372
- Syntax: How to Identify a FOCUS Database Server With USE. 372
- Using Report Procedures and a FOCUS Database Server. 373
- Accessing Report Procedures When Using a FOCUS Database Server. 373
- Procedure: How to Access Maintain and WebFOCUS Report Procedures on the Same Server. 374
- Reference: Communications Configuration File Location and Name. 375

#### Ensuring Transaction Integrity for DB2 Data Sources

- Reference: How WebFOCUS Maintain DB2 Logic Differs From Other IBI Products. 376

---

18 WebFOCUS
Using Transaction Locking to Manage DB2 Row Locks. ................................. 377
Procedure: How to Implement Transaction Locking for DB2. .................... 379
Using Change Verification to Manage DB2 Row Locks. ................................. 379
Procedure: How to Implement Change Verification for DB2. ........................ 380

14. Debugging WebFOCUS Maintain Applications ................................. 383
WebFOCUS Maintain Application Debugger ................................................. 383
Syntax: How to Invoke the WebFOCUS Maintain Application Debugger. ........... 383
Reference: Alternate Methods of Invoking the Maintain Application Debugger. . 384
Debugger Screen .................................................................................. 384
Debugging Your Application ................................................................ 385
Tracking Variables ........................................................................... 386
Navigation Control ............................................................................. 386
Procedure: How to Select a File and Find a Case ................................. 387
Reference: Error Messages and Type Statements ................................... 387
Reference: Usage Notes: Debugger Limitations .................................... 387
Type on EDAPRINT ............................................................................ 387
Maintain Statement Trace .................................................................. 388
Compiling Maintain Procedures ......................................................... 389
Enabling Server Tracing and Setting Trace Levels. .............................. 391
Reference: Examples of Enabling and Setting Tracing. ......................... 391
Viewing Trace Output ........................................................................ 392
Sample Usage Scenarios for Tracing ..................................................... 393
MNTCON PERFORMANCE_ANALYSIS ........................................... 393
Additional Trace Settings ................................................................. 393

A. Deploying Legacy Applications to the Web ................................. 395
What Are the Differences Between Windows-Deployed and Web-Deployed Applications? ................................. 395
Legacy Form Properties and Behavior ................................................. 395
Upgrading Legacy Windows Applications ......................................... 396

B. Glossary .................................................................................... 397

Customer Connections ..................................................................... 423

Developing WebFOCUS Maintain Applications 19
Preface

This documentation provides a reference for the WebFOCUS Maintain product. It is intended for application developers who are responsible for planning the enterprise software environment and for operating WebFOCUS Maintain.

How This Manual Is Organized

This manual includes the following chapters:

<table>
<thead>
<tr>
<th>Chapter/Appendix</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Setting Up Your WebFOCUS Maintain Project</td>
<td>Describes how to create, develop, and edit projects using the Maintain Development Environment.</td>
</tr>
<tr>
<td>2 Developing Procedures</td>
<td>Describes how to develop procedures, including passing parameters between procedures and using functions, variables, and data source stacks in procedures.</td>
</tr>
<tr>
<td>3 Using the Form Editor</td>
<td>Describes how to use the Form Editor to create forms.</td>
</tr>
<tr>
<td>4 Developing and Using Forms</td>
<td>Describes how to design and develop effective forms, how to manipulate their properties at design time and at run time, and how to integrate forms into your project.</td>
</tr>
<tr>
<td>5 Defining Events and Event Handlers</td>
<td>Introduces the different types of events, and describes how to use event handlers.</td>
</tr>
<tr>
<td>6 Developing and Using Controls</td>
<td>Introduces all controls for Maintain, and explains how to use them to create dynamic forms.</td>
</tr>
<tr>
<td>7 Form and Control Properties Reference</td>
<td>Describes all Maintain form and control properties.</td>
</tr>
<tr>
<td>8 Executing Other Procedures</td>
<td>Describes how to use the CALL and EXEC commands to execute other Maintain and WebFOCUS procedures, which is an important step to partitioning your application. Partitioning your application provides many advantages, including streamlined development and improved performance.</td>
</tr>
<tr>
<td>9 Using WebFOCUS Procedures in Your Application</td>
<td>Describes how to incorporate WebFOCUS procedures into your application.</td>
</tr>
</tbody>
</table>
### Documentation Conventions

The following table describes the documentation conventions that are used in this manual.

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>THIS TYPEFACE</strong> or <strong>this typeface</strong></td>
<td>Denotes syntax that you must enter exactly as shown.</td>
</tr>
<tr>
<td><strong>this typeface</strong></td>
<td>Represents a placeholder (or variable) in syntax for a value that you or the system must supply.</td>
</tr>
<tr>
<td><strong>underscore</strong></td>
<td>Indicates a default setting.</td>
</tr>
<tr>
<td><strong>this typeface</strong></td>
<td>Represents a placeholder (or variable), a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option that you can click or select.</td>
</tr>
</tbody>
</table>
## Convention

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key + Key</td>
<td>Indicates keys that you must press simultaneously.</td>
</tr>
<tr>
<td>{  }</td>
<td>Indicates two or three choices. Type one of them, not the braces.</td>
</tr>
<tr>
<td>[  ]</td>
<td>Indicates a group of optional parameters. None are required, but you may select one of them. Type only the parameter in the brackets, not the brackets.</td>
</tr>
<tr>
<td></td>
<td>Separates mutually exclusive choices in syntax. Type one of them, not the symbol.</td>
</tr>
<tr>
<td>...</td>
<td>Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis (...).</td>
</tr>
<tr>
<td>.</td>
<td>Indicates that there are (or could be) intervening or additional commands.</td>
</tr>
</tbody>
</table>

## Related Publications

Visit our Technical Content Library at [http://documentation.informationbuilders.com](http://documentation.informationbuilders.com). You can also contact the Publications Order Department at (800) 969-4636.

## Customer Support

Do you have any questions about this product?

Join the Focal Point community. Focal Point is our online developer center and more than a message board. It is an interactive network of more than 3,000 developers from almost every profession and industry, collaborating on solutions and sharing tips and techniques. Access Focal Point at [http://forums.informationbuilders.com/eve/forums](http://forums.informationbuilders.com/eve/forums).

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our website, [http://www.informationbuilders.com](http://www.informationbuilders.com). It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of [www.informationbuilders.com](http://www.informationbuilders.com) also provides usage techniques, diagnostic tips, and answers to frequently asked questions.
Call Information Builders Customer Support Services (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your questions. Information Builders consultants can also give you general guidance regarding product capabilities. Please be ready to provide your six-digit site code number (xxxx.xx) when you call.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

**Information You Should Have**

To help our consultants answer your questions effectively, be prepared to provide the following information when you call:

- Your six-digit site code (xxxx.xx).
- Your WebFOCUS configuration:
  - The front-end software you are using, including vendor and release.
  - The communications protocol (for example, TCP/IP or HLLAPI), including vendor and release.
  - The software release.
  - Your server version and release. You can find this information using the Version option in the Web Console.
- The stored procedure (preferably with line numbers) or SQL statements being used in server access.
- The Master File and Access File.
- The exact nature of the problem:
  - Are the results or the format incorrect? Are the text or calculations missing or misplaced?
  - Provide the error message and return code, if applicable.
  - Is this related to any other problem?
  - Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?
  - What release of the operating system are you using? Has it, your security system, communications protocol, or front-end software changed?
  - Is this problem reproducible? If so, how?
Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two data sources, have you tried executing a query containing just the code to access the data source?

Do you have a trace file?

How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?

User Feedback

In an effort to produce effective documentation, the Technical Content Management staff welcomes your opinions regarding this document. You can contact us through our website, http://documentation.informationbuilders.com/connections.asp.

Thank you, in advance, for your comments.

Information Builders Consulting and Training

Interested in training? Information Builders Education Department offers a wide variety of training courses for this and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our website (http://education.informationbuilders.com) or call (800) 969-INFO to speak to an Education Representative.
You can use the Maintain Development Environment to design, test, and deploy WebFOCUS Maintain projects.

The following topics describe how to open the Maintain Development Environment and how to create WebFOCUS Maintain projects (the umbrella under which the files of the application are stored). These topics also describe the basic building blocks of a project and the various parts of the Maintain Development Environment.

For detailed information on deployment, see Partitioning and Deploying Project Files in the Creating Reporting Applications With Developer Studio manual.

For information on using the Update Assist Wizard to rapidly create WebFOCUS Maintain applications, refer to the Creating Reporting Applications With Developer Studio manual.

**In this chapter:**

- Opening WebFOCUS Maintain
- Applications and Projects
- What Are the Building Blocks of a WebFOCUS Maintain Project?
- Editing Projects in the Maintain Development Environment
- Viewing System Messages in the Output Window
- Searching for Text in Your Project
- Editing Project Components as Text
- Incorporating Data Sources Into Your Project
- Using the Resource Wizard
- Team Development

**Opening WebFOCUS Maintain**

WebFOCUS Maintain is an optional part of WebFOCUS Developer Studio, which you can open from the main Developer Studio window.
Procedure: How to Open the Maintain Development Environment

To open the Maintain Development Environment, do one of the following:

- In Developer Studio, click the Open MAINTAIN button.

  or

- Open the Maintain Files folder for any project and double-click on a procedure.

Applications and Projects

An application is a group of procedures and other components that work together to achieve a common goal, such as an online ordering system (like an e-commerce site). An application can be distributed across many servers, access varied data sources, and support multiple users.

When you are ready to begin developing an application, your first step is to create a WebFOCUS Maintain project. A project is the highest level object that you can create with WebFOCUS Maintain. When you deploy your project to a WebFOCUS Server, your project becomes an application.

Procedure: How to Create a Project

You can create WebFOCUS Maintain projects in both WebFOCUS Developer Studio and the Maintain Development Environment. The procedure is the same:

1. Right-click Projects on localhost in the Explorer, and then click New Project.

2. In the Create a Project – Step 1 of 2 dialog box, enter the name of your project and the directory where you want to save it. This name must begin with a letter and contain no special characters (but it can include spaces).

   By default, the directory name is the same as the project name. You can typically find this directory in \ibi\apps folder (also known as the APPROOT folder).

3. Click Next.

4. If you specify the name of a directory that does not exist, Developer Studio asks if you want to create it. Click Yes.

5. In the Create a Project – Step 2 of 2 dialog box, enter any directories that contain files that you want to use (this step is optional, and you can always add new directories later using the Properties dialog box).

6. Click Finish.

Developer Studio creates a briefcase for your project in the Explorer window.
What Are the Building Blocks of a WebFOCUS Maintain Project?

After you open the Maintain Development Environment and create a project, you must add components to your project. You can add any file you want to your project, although when you deploy your project to create an application, these files should make sense in the context of where you are deploying them. For example, WebFOCUS Maintain can deploy to MVS machines, but deploying a graphic file there may not work well.

When you first create a project, WebFOCUS Maintain creates four folders that represent some common components that you may want to place in your project:

- **HTML Forms.** Files with the extensions .htm and .html.
- **Maintain Files.** Maintain procedures and forms. You create them using the Maintain Development Environment. For more information on procedures, see Developing Procedures on page 55.
- **Master Files.** Files that describe the data on which WebFOCUS Maintain operates and have the extensions .mas and .acx. For more information, see Incorporating Data Sources Into Your Project on page 43.
- **Procedures.** New FOCUS procedures you create using WebFOCUS Developer Studio. For more information, see Using WebFOCUS Procedures in Your Application on page 315.

You can add components to a project either by creating new ones or by copying existing ones.

Creating New Project Files

You can use WebFOCUS Maintain to create new project files for the following items:

- Maintain procedure
- (Web)FOCUS procedure
- Master File
- VBScript
- JavaScript®
- HTML document
- Text

**Procedure:** How to Create a Project Component

To create a project component:
What Are the Building Blocks of a WebFOCUS Maintain Project?

1. Do one of the following:
   - Right-click either the project or one of its folders in the Explorer, click New in the pop-up menu, and then click Project file.
   - or
   - In the File menu, click New, then click Project file.

2. In the Create new project file dialog box, select the type for the new component, as shown in the following image.

3. Choose a location for the component. Your options include any of the directories in the project path. For more information on paths, see How to Change the Project Path on page 34.

4. Type a name for the component. This name must conform to Windows naming standards. WebFOCUS Maintain appends the appropriate extension to the file name.

5. If you are creating a Maintain procedure, you can select the Import module check box to indicate that the procedure will be an import module. An import module is a procedure that contains functions that are called by other Maintain procedures only, but does not run on its own. This means that there will be no Top function. For more information on import modules, see Developing Procedures on page 55.

6. Click OK.
Adding Existing Files to Your Project

In addition to creating new files, you can also add existing files to your project. This option allows you to reuse existing files without having to recreate them.

Procedure: How to Add an Existing File to Your Project

To add an existing file to your project:

1. Ensure that WebFOCUS can see the type of file you want to add to the project. If the file is an HTML File (.htm or .html), a Maintain procedure (.mtn or .for), a Master file (.mas or .acx), or a FOCUS procedure (.fex), you do not need to do anything here. If you are adding another type of file, see How to View New File Types in the Explorer on page 35.

2. Ensure the file you want to add is in the project path. Do one of the following:
   - Open the Properties dialog box for the project, click the Directories tab, and add the directory where the file is located.
   - Using Windows Explorer, move the file to one of the directories that are in the project path.

3. Turn on Display all files in the project paths:
   - Click the Display all files in the project paths button.
   - In the View menu, turn on Show All Files.

4. Right-click the file in the Explorer and click Add to Project in the pop-up menu.

Procedure: How to Add an Existing File From Another WebFOCUS Server to Your Project

If the file you want to add to your project is not on your machine, but located on another WebFOCUS Server, you can still add it to your project.

1. In the Explorer window, expand Remote Application Servers.
2. Expand the WebFOCUS Server where the file you want to add to your project is located.
3. When you find the file, drag it into your project folder.
What Other Types of Files Can Go Into a WebFOCUS Maintain Project?

There are many other kinds of files that you might want to include in your WebFOCUS Maintain project.

For example, you can include graphics, ActiveX controls, and any of the following types of external procedures:

- 3GL programs (COBOL, C, C++).
- CICS transactions.
- IMS/TM transactions.
- RDBMS stored procedures (from Oracle® or Sybase®, for example).

Being able to include all of these kinds of procedures into your project can preserve your investments in these pieces of software.

- To execute an external procedure with static parameters from a Maintain procedure, you use the EXEC command to call a WebFOCUS procedure that will execute the external procedure. For more information, see Executing Other Procedures on page 297, and Using WebFOCUS Procedures in Your Application on page 315.

- You can also execute external procedures from a script function using the IWCLink function. This enables you to pass dynamic parameters to the external procedure. For more information, see the Using Functions manual.

Defining the Flow of a Project

A project must contain one or more procedures. One of these procedures is your starting procedure, which is executed first when you execute the project. The other procedures are executed only if they are called explicitly using the CALL syntax.

A procedure consists of functions (defined by the CASE and ENDCASE keywords). One of the functions in your procedure is called Top, and is executed when the procedure is executed. As with procedures, the other functions are executed only if they are called explicitly using the PERFORM syntax.

To display an initial form somewhere in the Top Case of the starting procedure, you must insert the command that displays this form (or Top Case should call another function or procedure that executes this command). The command you use to display a form is Winform Show form.

Before setting up a project, it is recommended that you read Designing Applications for Scalability and Performance on page 351.
Editing Projects in the Maintain Development Environment

When you open a new WebFOCUS Maintain project, your Maintain Development Environment looks similar to the following image.

By default, you see the following two windows (depending on which windows you have open and what settings you have changed, your environment may look slightly different):

**Explorer**

This window is where you manage your projects, their components, and the servers you can access. This window is usually on the right.

For more information on using this window, see the Maintain Getting Started manual.

**Project Explorer**

Contains another view of the Maintain procedures in your project. This window also has a Display all files in the project paths button. This window is usually on the left.

If the project and any other component in it have asterisks (*) next to their names, you know that you have made changes to that component since the last save.
Procedure: How to Save Your Project or a Component in Your Project

When a project component has been modified since the last time you saved it, you see an asterisk (*) next to the name of your project and the name of the component. To save a component in your project:

1. Select the component you want to save.
2. In the File menu, click Save.
   
   or

   Click the Save button on the General toolbar.

To save the entire project, do one of the following:

- In the File menu, click Save all.

   or

- Click the Save all button on the General toolbar.

Procedure: How to Save Your Project Automatically

To automatically save your project:

1. In the Tools menu, click Environment options.
2. Select the Automatically save projects every n minutes check box.
3. Enter how often you want to save your projects, or leave the default set to five minutes.
4. Click OK to confirm your changes.

Note: The default for this option is on.

Procedure: How to Change the Project Path

By default, the path of the project includes the directory IBI\apps\projectname, but you may want to add new directories to this path.

1. In the Explorer window, right-click the project, and then click Properties.
2. In the Properties dialog box, click the Directories tab.
3. You can do any of the following:
To add a new directory, click the **New** button to open the Browse for folder dialog box, select a directory you want to add, and click **OK**.

**b.** To remove a directory from the path, select it and click the **Delete** button.

**c.** To change the search order of a directory, select it and use the **Move item up** button or the **Move item down** button.

4. Click **OK** when you are done.

**Procedure: How to View New File Types in the Explorer**

By default, you can see HTML files (.html, .htm), Maintain files (.mtn, .for), Master files (.mas, .acx), and Procedures (.fex) in the Explorer.

1. In the Explorer window, right-click the project, and then click **Properties**.

2. In the Properties dialog box, click the **Edit Filters** tab.

3. Click the **Add new file type filter(s)** button.

4. In the New filter dialog box, select the file type you wish to add and click **OK**.

5. In the Properties dialog box, click **OK**.

The WebFOCUS Maintain Development Environment can now display files of this type.

**Note:** You can customize the folders in a project to display various file types. For more information, see the *Creating Reporting Applications With Developer Studio* manual.

**Procedure: How to View or Hide All Files in the Project Path**

To show or hide files in the path of the project:

- Click the **Display all files in the project paths** button.

  or

- In the View menu, click **Show All Files**.

**Procedure: How to Rename a Project or Project Component**

To rename a project or component:

1. Right-click the project or project component and click **Rename**.
2. Type a new name and press Enter.

**Procedure: How to Delete a Project or Project Component**

To delete a project or component:

1. Right-click the project or project component and click *Delete*.
2. Confirm that you want to delete the project or project component.

**Note:**

- Deleting a project component is different from removing it from the project.
- If you delete a project this way, WebFOCUS Maintain deletes the .gfa file in the project folder, but does not delete the folder in the Windows file system.
- If you delete a project component this way, WebFOCUS Maintain deletes the corresponding file from your hard drive.

**Procedure: How to Remove a Project Component From a Project**

To remove a project component from a project:

- Right-click the project component and click *Remove from project*.

**Note:** The project component still exists, so you can add it back to the project, if you wish.

**Viewing System Messages in the Output Window**

The Output Window displays messages generated by WebFOCUS Maintain. These messages are split into three different types, represented by tabs at the bottom of the window:

**General**

Shows all messages that have to do with the project in general, such as opening a project or checking in or checking out a part of a project.

**Find**

Displays search results from the Search project command and enables you to open those project components that satisfy your search conditions.
An example of the Output Window is displayed in the following image.

![Output Window](image)

**Procedure:**  How to Open and Close the Output Window

In the View menu, click *Output window*.

**Searching for Text in Your Project**

You can search for text throughout your entire project, using the Search project command, as shown in a following image. You can also search for text in an individual project component, such as a procedure, while editing as text. For more information, see *How to Find Text* on page 41.

![Search project](image)

WebFOCUS Maintain displays the results of your search in the Find tab of the Output window.

**Procedure:**  How to Search for Text in Your Project

To search for text in your project:

1.  In the Edit menu, click *Search project*. 
Click the Search project button on the General toolbar.

2. In the Search project dialog field, enter the text for which you want to search in the Find What field or use the text from a previous search by selecting it from the list.

3. Select the file type from the In file types list.

4. Select the project from the In project list.

5. If you want to match the case of the text for which you are searching, select Match case.

6. If you want to match only whole words, select Match whole word only.

7. Select Look in subfolders to search for text in the subfolders.

8. Select Search all application paths to look for text in an application path.

9. Click Find Now.

10. If the Output window is not open, open it by clicking Output window in the View menu.

11. WebFOCUS Maintain displays a list of the project components that contain the text in the Find tab of the Output window. To open a component at the instance of the text, double-click its line in the Find tab.

Editing Project Components as Text

WebFOCUS Maintain provides the following text editors:

- The Procedure Editor enables you to edit the Maintain language source code for a procedure.
- The Script Editor enables you to edit the JavaScript or VBScript code for a script library.
- The WebFOCUS Procedure Editor enables you to edit the code for a WebFOCUS procedure (although it is recommended that you use WebFOCUS Developer Studio to edit your WebFOCUS procedures).
- The Master File Editor enables you to edit the code for a Master File, which describes the structure of a data source.
- The HTML File Editor enables you to edit the code for an HTML file.
- The Event Handler editor enables you to edit the Maintain, JavaScript, or VBScript code for an event handler.

You can open and edit code or other text for multiple components in separate editor windows.
**Procedure:** How to Open a Project Component as Text

In the Project Explorer, right-click a component and select *Edit* or *Edit source* from the shortcut menu.

The editor window displays the underlying code or text for the selected project component.

**Customizing Parsing Activity**

Changes to code in the editor are parsed and automatically incorporated in the corresponding graphical application component.

Developers can now control the lag time for the parser of the editor by customizing how long the parser waits for the developer to stop typing before it automatically parses the Maintain code. Developers can also disable the parser completely to optimize the Maintain Development Environment when they are in a typing-intensive session. They can re-enable the parser to update the Project Explorer tree automatically.

**Procedure:** How to Customize Parsing Activity

To customize parsing:

1. Change or adjust the parsing option for the Maintain Development Environment (MDE) by selecting *Tools*, and then *Environment Options*.

   MDE displays the Environment Options dialog box.

2. On the General tab, check *parse editor* to enable parsing of your code in the code Editor, or uncheck it to disable code parsing.

   If the parser is enabled, it waits until you stop typing before it begins to parse your code for changes that might affect the Project Explorer.

   If you disable parsing, changes to your code are not reflected in the Project Explorer until you save them.

3. Use the spin button to adjust the amount of time (in seconds) the parser waits before starting to parse your code or type the desired wait time into the box.

**Common Editing Functions**

The following is a list of common editing functions that you can perform.

**Procedure:** How to Enter Text

To enter text, place the cursor where you want the text to go and start typing.
Editing Project Components as Text

**Note:** The editor does not wrap text automatically, so when you reach the end of a line, you must press Enter.

**Procedure:** How to Select a Range of Text

To select a specific range of text:

1. Position the cursor at the beginning of the text you want to edit.
2. Press the left mouse button and drag so that the pointer passes over all the text you want to select.
3. When the pointer reaches the end of the text, release the left mouse button.

**Procedure:** How to Select All Text

To select all text, do one of the following:

- In the Edit menu, click **Select all**.
- Press Ctrl+A.

**Procedure:** How to Move Text to Another Location

To move text:

1. Select the text you want to move.
2. Use the Cut and Paste commands, or drag the text to its new location.

**Procedure:** How to Copy Text to Another Location

To copy text:

1. Select the text you want to copy.
2. Use the Copy and Paste commands, or hold down the Ctrl key and drag the text to its new location.

**Procedure:** How to Undo or Redo an Action

Do one of the following:

- In the Edit menu, click **Undo or Redo**.

- Click the **Undo** button or the **Redo** button on the General toolbar.
Press Ctrl+Z to undo, or Ctrl+Y to redo.

**Procedure: How to Find Text**

To locate specific text:

1. On the Edit menu, click *Find*, or click the *Find* button on the Editor toolbar to open the Find dialog box.
2. In the *Find What* field, type the word or phrase you want to find.
3. Select any search criteria that help to narrow or speed your search.

   **Note:** You can create some powerful searches using regular expression syntax. For more information, try searching the web.

4. Click *Find Next* to begin searching for the specified text in the open editor window. The text appears highlighted.

5. Click *Find Next* in the Find dialog box or press F3 to search for another occurrence of the text.

**Procedure: How to Replace Text**

To replace specific text:

1. On the Edit menu, click *Replace* to open the Replace dialog box.
2. In the Find What field, type the word or phrase you want to replace.
3. In the Replace With field, type the new word or phrase with which you want to replace the old word or phrase.
4. Select any search criteria that help to narrow or speed your search.

   **Note:** You can create some powerful searches using regular expression syntax. For more information, try searching the web.

5. Click *Find Next* to begin searching for the specified text.

   WebFOCUS Maintain finds and highlights the text.

6. Click *Replace* to replace the highlighted text with your new text or phrase.

7. WebFOCUS Maintain automatically highlights the next occurrence of the text. Click *Replace*, if you wish to repeat the substitution.
How to Preview an Application Component as Text

Procedure:
In the File menu, click Print preview.

The text appears on the screen as it will look when printed.

How to Print an Application Component as Text

Procedure:
To print, do one of the following:

- In the File menu, click Print.

or

- Click the Print button on the General toolbar.

Using Bookmarks to Move Within the Editor

The bookmark option places a bullet next to or removes a bullet from any line in the editor. You can place multiple bookmarks in a file by using the Mark All function in the Find dialog box. After a bookmark is added, you can jump to that bookmark from anywhere in a file. If you have multiple bookmarks, the cursor jumps to the very next bookmark.

Bookmarks are temporary. They disappear when you close the editor window.

How to Apply or Remove a Bookmark

Procedure:
To add or remove bookmarks:

1. Position the cursor anywhere in a line where you want the bookmark to appear.
2. Right-click, then select Toggle Bookmark.

or

Click the Toggle Bookmark button on the Editor toolbar.

or

Press Ctrl+F2.
Procedure: How to Jump to a Bookmark

To jump to another bookmark, click the Next Bookmark button or Previous Bookmark button.

The cursor jumps to the next or previous bookmark in the editor.

Procedure: How to Apply Multiple Bookmarks

To apply multiple bookmarks:

1. On the Edit menu, select Find.
   
   The Find dialog box opens.

2. In the Find What field, type the word or phrase you want to find, then click Mark All to mark all lines with this text.

Procedure: How to Remove All Bookmarks

To remove all the bookmarks, do one of the following:

- Click the Clear All Bookmarks button on the Editor toolbar.
  
  or

- Close the editor window and reopen it.

Incorporating Data Sources Into Your Project

One of the most frequently used components in a WebFOCUS Maintain project is a data source description, also known as a Master File. The data source description describes the structure of a data source where you store the data for your project. For example, the data for a human resources project would be the employee names, addresses, social security numbers, and so on. For an online-ordering system, the data would be the available products, their prices, their shipping costs, and so on. WebFOCUS Maintain cannot read the information in the data source without the data source description telling it how the data is organized.
A data source description is stored in two separate files:

- The Master File, with the suffix .MAS, describes the structure of the data.
- The Access File, with the suffix .ACX, describes how to access the data. If your data source is local and is in either FOCUS or text format, an Access File is not necessary.

WebFOCUS refers to data source descriptions as Master Files.

The data source itself is never part of your project. One reason for this is for data integrity purposes. If you make the data source part of your project and deploy it to different locations, you will have multiple copies of the same data, and it will be difficult to determine which one is correct.

Another reason is that the stored data source is platform-dependent. You cannot copy binary data source files from one platform to another and expect them to be readable on the new platform.

**Note:** If you use an existing data source description and there is a corresponding Access File, WebFOCUS Maintain includes the Access File in the project with the Master File. You cannot see or edit this Access File, but WebFOCUS Maintain deploys it with the Master File.

### Procedure: How to Edit a Data Source Description From the MDE

To edit a data source description from the MDE:

1. Right-click the data source in the Project Explorer, or in the Object Explorer within the MDE.
2. In the shortcut menu, click **Open**.

WebFOCUS Maintain opens the data source in a text editor. Although you can make changes here, they are not registered by the MDE until you exit and reenter.

It is highly recommended that you use the Developer Studio Synonym Editor or another editor to edit your Master File before starting your MDE development session.

### Procedure: How to Edit a Data Source Description From Developer Studio

To edit a data source description from Developer Studio:

1. Go to the Object Explorer in your Developer Studio session and right-click the data source.
2. In the shortcut menu, click **Open**.

The data source opens in the Synonym Editor. For more information, see the *Describing Data With Graphical Tools* manual.
Note: Editing the data source description does not change the actual data in the data source.

Procedure: **How to Supply a WebFOCUS DBA Password**

If you are developing a WebFOCUS Maintain application using a data source description that uses DBA, you must specify the DBA password so that WebFOCUS Maintain can access this database.

1. In the Edit menu, click **Password**.

   The Password Dialog box opens, as shown in the following image.

   ![Password Dialog](image)

2. Enter a password in the dialog box and click **OK**.

Procedure: **How to View the Structure of a Data Source**

You can only view the structure of a data source in the Project Explorer, and a data source does not appear in the Project Explorer unless you use it in a procedure. For more information, see *Specifying Data Sources for Your Procedure* on page 58.

1. In the Project Explorer, click the plus sign (+) to the left of the data source name.

   WebFOCUS Maintain displays the names of the segments in the data source.

2. Click the plus sign (+) to the left of the segment name.

   WebFOCUS Maintain displays the names of the fields in the data source. Any key fields have a key to the left of them.
**Example:**  **Viewing the Structure of VIDEOTRK**

The following window shows what happens when you expand the data source description for VIDEOTRK, one of the sample data sources.

VIDEOTRK has a segment named CUST, with the fields CUSTID, LASTNAME, FIRSTNAME, and so on. CUSTID is a key field.

**Using the Resource Wizard**

The Resource Wizard helps you add resources, such as image files, HTML files, Java applets, and ActiveX controls to your project. Resources are non-native Maintain project components that you can use in the forms you design.
References: **Resource Wizard: Specify Path for Image**

Use the dialog box shown below to specify a path to an image.

This dialog box includes the following options:

**Image path**

Displays the location of your image file by its full path and file name. If you wish, you can type a path and file name here.

Opens the Open dialog box, where you can search for your image file.

**Preview**

Enables you to view an image whose path you just typed.
Reference: Resource Wizard: Specify Path for HTML File

Use the dialog box shown below to specify a path to an HTML file.

This dialog box includes the following options:

**HTML file path**

Displays the location of your HTML file by its full path and file name. If you wish, you can type a path and file name here.

Opens the Open dialog box, where you can search for your HTML file.

**Refresh**

Enables you to view an HTML file whose path you just typed.
Reference: Resource Wizard: Specify Path for Java Applet

Use the Resource Wizard dialog box to specify a path to a Java applet, as shown in the following image.

This dialog box includes the following options:

**Applet location**

Displays the location of your Java applet by its full path and file name. If you wish, you can type a path and file name here.

**Note:** If your Java applet involves using other Java classes, you must either package the applet with these classes, or inform your end users to include the location of these classes in their CLASSPATH. For more information, see the documentation for Internet Explorer.

Opens the Open dialog box, where you can search for your Java applet.
**Description**
Enter a description of your Java applet.

**Reference:**  **Resource Wizard: Define Parameters for Java Applet**
Java applets often have parameters for which you can specify values, thus affecting the behavior of the Java applet in your application. Usually, you can find out the names and default values of these parameters in a text file that accompanies your Java applet.

After you know the names of your Java applet parameters, you can define them in this dialog box. You only need to define the parameters you want to use in your application.

This dialog box includes the following options:

- **Opens the Enter a Parameter dialog box, where you can define parameters and their default values.**

- **Deletes the selected parameter.**
Moves a selected parameter up in the list of parameters.

Moves a selected parameter down in the list of parameters.

**Reference:** Resource Wizard: Specify ActiveX Controls

The Resource Wizard, shown in the image below, searches through all of the registered ActiveX controls on your computer and displays them in the Insert ActiveX Control dialog box. When you select an ActiveX control from the list in the Insert ActiveX Control dialog box, it appears on the screen by default. If the ActiveX control you want does not appear on the list, you must download it to your computer.

![Resource Wizard](image)

The Resource Wizard dialog box allows you to browse registered controls on your computer and select a control to insert. The selected control appears on the screen by default. If the control is not visible, you need to download it to your computer.
This dialog box includes the following options:

**Browse for registered controls on My Computer**

Opens the Insert ActiveX Control dialog box so that you can insert an ActiveX control to use in your application.

**Control's name**

Displays the name of the ActiveX control selected from the Insert ActiveX Control dialog box.

**Where the control is installed on your computer**

Displays where the ActiveX control resides on your computer.

**Reference:**  Open Dialog Box

Use the Open dialog box, shown in the following image, to browse for a file to add to your project. This dialog box is very similar to a standard Windows Open dialog box, but with one important difference. It enables you to browse through any WebFOCUS or web servers you can access.

![Open Dialog Box](image)

This dialog box includes the following options:

**Look in**

Contains the directory where you are browsing for files. If you look in the Developer Studio Desktop, you can switch from looking through the Windows file system to looking through your remote application servers.
Remote Application Servers

Double-click here to browse through any WebFOCUS Servers or web servers you can access if you are looking in the Developer Studio Desktop.

Windows Desktop

Double-click here to browse through the files on your computer if you are looking in the Developer Studio Desktop.

Reference: Resource Wizard: Specify a Unique Name

When you add a resource to your project, you must specify a unique name for your resource file in Resource Wizard, shown in the following image. WebFOCUS Maintain refers to this name when deploying your project. This prevents you from overwriting resource files with the same name from other projects.

Resource names must begin with a letter and can include a combination of letters, numbers, and underscores. Names can consist of a maximum of 66 characters.
**Reference: Copy Resource Dialog Box**

This dialog box, shown in the image below, appears when you are adding an existing file, such as an HTML file, image, or Java applet, as a resource to your project, using the Resource Wizard. This dialog box tells you that it is creating a copy of the resource in your project directory. At deployment time, WebFOCUS Maintain ensures that the resource gets deployed to the proper location. Select the *Don’t show this again* option if you do not need WebFOCUS Maintain to display this dialog box.

![Copy Resource Dialog Box](image)

This dialog box also warns you if you are overwriting a resource with the same name, as shown in the following image. You can either rename the resource or overwrite the existing one.

![Resource Overwrite Dialog Box](image)

**Team Development**

WebFOCUS Maintain fully supports team development by enabling developers to source manage their application components. For more information, see *Creating a Reporting Application* in the *Creating Reporting Applications With Developer Studio* manual.
Developing Procedures

Procedures are multipurpose project components that accomplish specific tasks. They are the building blocks of a project and they can access data sources, call other procedures, and transfer data.

This topic provides information on creating, editing, and viewing procedures. It also contains information on specifying data sources to access in a procedure, using functions and variables in procedures, using data source stacks to manipulate information in other data sources, and using the Type Wizard to assign data types.

In this chapter:
- Overview
- Specifying Data Sources for Your Procedure
- Using Functions in Procedures
- Using Variables in Procedures
- Using Data Source Stacks in Procedures
- Using the Type Wizard
- Using Import Modules
- Testing Procedures With the Run Procedure Option

Overview

When you develop your project, you divide it into a few logical components that accomplish specific tasks, such as placing a new order or checking the status of an existing order. These logical components are called procedures. Procedures are the main building blocks of your project.

For information on how to create a procedure, see How to Create a Project Component on page 29. For information on how to add an existing procedure to your project, see How to Add an Existing File to Your Project on page 31.

Procedures can operate on data sources. If a procedure accesses a data source, either to extract data or to write data, you must specify which data sources the procedure accesses. For more information, see Specifying Data Sources for Your Procedure on page 58.
A procedure can consist of any combination of the following:

- **Functions.** For more information, see *Using Functions in Procedures* on page 60.

- **Variables.** For more information, see *Using Variables in Procedures* on page 67.

- **Forms.** For more information, see *Developing and Using Forms* on page 109.

- **Data Source Stacks.** For more information, see *Using Data Source Stacks in Procedures* on page 72.

- **Classes.** For more information, see *Developing Classes and Objects* on page 327.

 Procedures can also call other procedures as a part of their execution, and can pass data back and forth using input and output parameters. For more information, see *Executing Other Procedures* on page 297.

A special kind of procedure is called an *import module*. An import module is a non-executable procedure that contains Maintain language code. You do not execute an import module as part of your project. Instead, you use it as a library of functions or classes.

**Procedure: How to Designate the Starting Procedure**

One of your procedures is designated as the *starting procedure*. When you execute the project, WebFOCUS Maintain executes the starting procedure first.

1. Open the Properties dialog box for the project.
2. In the Starting Object list box, select the procedure you wish to use as your starting procedure.
3. Click OK.

**Procedure: How to Edit the Source Code of a Procedure**

WebFOCUS Maintain generates Maintain language code for all procedure components. You can edit this code directly in the Procedure Editor.

1. Right-click the procedure.
2. In the shortcut menu, click *Edit source* or *Open*.
3. All procedures begin with the keyword `MAINTAIN` and end with the keyword `END`. Make your desired changes to the code between these two keywords.
   
   For more information, see *Command Reference* in the *Maintain Language Reference* manual.

4. Close the Procedure Editor.
**Procedure: How to Use the Language Wizard**

WebFOCUS Maintain provides a Language Wizard to help you construct Maintain language source code.

1. Place the cursor where you want your source code to go.
2. Right-click at the insertion point and choose *Language Wizard* from the shortcut menu. The first screen of the Language Wizard opens.
3. Select the type of code you wish to create and follow the online instructions.

The Language Wizard can generate Maintain commands to perform the following tasks:

- Retrieve records from a data source (NEXT).
- Update records in a data source (INCLUDE/UPDATE).
- Operate on a form (Winform).
- Run another procedure (CALL or EXEC).
- Operate on a stack (STACK).

**Procedure: How to Access Help for Any Maintain Language Keyword**

1. Select the keyword.
2. Press the F1 key.

**Procedure: How to View the Description of a Procedure Component**

When you create a function, variable, class, or data source stack using the Function Editor, Variable Editor, Class Editor, or Stack Editor, the Description tab enables you to enter a description for that component. To view this description:

1. Select the function, variable, class, or data source stack.
or
Right-click the function, variable, class, or data source stack, and in the shortcut menu, click Show description.

**Procedure: How to View Your Procedure Components in Folders**

By default, your functions, variables, stacks, and classes appear directly under your procedure in the Project Explorer, while your forms appear in a Forms folder under the procedure.

If you wish, you can display the functions, variables, data source stacks, and classes in their own folders under the procedure:

1. In the Tools menu, click Environment options.
2. Click the Project Explorer tab.
3. Select the procedure components that you want to display in folders.
4. Click OK.

**Specifying Data Sources for Your Procedure**

If your procedure accesses a data source, either to display data or to enter information into the data source, you must tell the procedure which data sources to access from the Data Sources folder of the project.

WebFOCUS Maintain creates a Data Sources folder under your procedure and lists the data sources your procedure is using.

**Note:** If you edit your procedure as text, specify the data sources to use in your procedure after the MAINTAIN FILE keywords. For more information, see the *Maintain Language Reference* manual.

If you specify a data source that is not in the list of data sources in the project, WebFOCUS Maintain opens the Unknown Reference Wizard. For more information, see *Setting Up Your WebFOCUS Maintain Project* on page 27.

**Procedure: How to Specify Data Sources for Your Procedure**

1. Right-click the procedure.
2. In the shortcut menu, click Use data sources.
   The Use These Data Sources in Procedure dialog box opens.
3. Select the data sources you want to use in the *Available data sources* list and click the right arrow button [►]. You can display or hide all the data sources in the project paths by clicking the *Display all files in the project paths* button [.Visible].

**Tip:** If you do not see the data source you want to use, you must ensure it is in the project paths. For more information, see *How to Add an Existing File to Your Project* on page 31.

4. To remove data sources you no longer need, select them in the *Data sources to use* list and click the left arrow button [◄].

5. Use the *Move down* and *Move up* buttons to change the order of the data sources (the only time order matters is when data sources have identical field names and you refer to one without a qualifier).

6. Click *OK*.

If you select a data source that is not part of the project, WebFOCUS Maintain adds it to the project.

**Reference:**  *Use These Data Sources in Procedure Dialog Box*

Specify the data sources to use in your procedure with the Use these Data Sources in Procedure dialog box, as shown in the following image.

![Use these Data Sources in Procedure Dialog Box](image)

This dialog box includes the following options:

**Available data sources**

Contains a list of the data sources in your project paths.
Displays or hides the data sources in the project paths that are not part of the project.

Copies a selected data source into the list of data sources to use.

Removes a selected data source from the list of data sources to use.

**Data sources to use**

Contains a list of the data sources this procedure uses.

Moves a selected data source up in the list of data sources.

Moves a selected data source down in the list of data sources.

**Using Functions in Procedures**

A function, also known as a case, is a series of commands in a procedure grouped together as a unit of control. A function accomplishes some small task, such as calculating values, extracting data from a data source to place in a data source stack, or writing information to a data source.

All executable code must be in a function. The only types of logic allowed outside of a function are variable declarations, object declarations, and class definitions (CASE, DECLARE, DESCRIBE, END, MAINTAIN, and MODULE statements).

Some functions have arguments, meaning when you invoke the function, you must supply information that the function needs to complete its task.

Some functions also have a return value, meaning they calculate a value and return this value to whatever called the function. Functions that are event handlers (that is, that are invoked by some action in a form) cannot have return values.
Notice that local variables (defined in a function using the DECLARE command) are available only within that function. As soon as the function completes processing, these values are no longer available. Global variables (defined using the DECLARE command outside of a function or using the COMPUTE command anywhere) are available for the entire procedure. For more information on local and global variables, see the Maintain Language Reference manual.

You can also call one function from within another function.

The combination of the arguments and return value of a function is known as its signature.

**Procedure:** How to Create a Function

1. Select the procedure that you want the function to be part of.
2. Right-click the procedure, click New in the shortcut menu, and click Function (Case) in the submenu.
   
   or

   Click the New function button on the Maintain Objects toolbar.
3. In the New Function dialog box, type a name for your function in the Name field (up to 66 characters with no spaces).
4. If your function requires any arguments, use the New button to open the Function Parameter dialog box to add them (not all functions require arguments).
5. If your function requires a return value, enter its name and format in the Returns field, or click the ellipsis button to open the Function Return dialog box. This is where you can define the function.
6. If you wish, click the Description tab and type a description.
7. Click OK.
8. Right-click the function and select Edit Source from the shortcut menu.

   WebFOCUS Maintain opens the procedure containing your function in the Procedure Editor at the code for the function.
9. Locate your cursor just before the ENDCASE keyword and enter the Maintain language commands that determine what this function does.

    **Note:** The following Maintain language commands are not valid within the context of a function: CASE, ENDCASE, MAINTAIN, END, MODULE, and DESCRIBE.
Tip: You can add Maintain language code to your function using the Language Wizard. Place your insertion point where you want your code to go, right-click in the Procedure Editor window, and in the shortcut menu, click Language Wizard.

You can obtain Help on any Maintain language keyword by selecting the keyword and pressing F1 key.

10. Close the Procedure Editor.

**Procedure:** How to Edit the Name, Arguments, Return Value, and Description of a Function

1. Right-click the function in the Project Explorer.
2. In the shortcut menu, click Edit.
3. Make your desired changes in the Edit Function dialog box.
4. Click OK.

**Procedure:** How to Edit a Function

1. Right-click the function in the Project Explorer.
2. In the shortcut menu, click Edit Source.
3. Make your desired changes to the code between the CASE functionname and END_CASE keywords.

For more information, see the Maintain Language Reference manual.

4. Close the Procedure Editor.

Tip: You can add Maintain language code to your function using the Language Wizard. Place your insertion point where you want your code to go, right-click in the Procedure Editor window, and in the shortcut menu, click Language Wizard.

You can obtain Help on any Maintain language keyword by selecting the keyword or placing your insertion point in the keyword and pressing F1 key.

**Procedure:** How to Move a Function to Another Procedure

1. In the Project Explorer, ensure that you can see both the function and the procedure to where you want to move it.
2. Select the function and drag it to the procedure to where you want to move it.
Procedure: How to Copy a Function to Another Procedure

1. In the Project Explorer, ensure that you can see both the function and the procedure to where you want to move it.

2. Select the function, hold down the Ctrl key, and drag the function to the procedure to where you want to move it.

Reference: Function Editor: Signature Tab

When you create or edit a function, WebFOCUS Maintain opens the New Function, Edit Function, or Member Function dialog box, as shown in the following image.

This dialog box includes the following options:

Name

The name you assign to your case. This name can be up to 66 characters long. It must begin with a letter and can include any combination of letters, digits, and underscores (\_).

Takes: (Name/Type)

Lists the arguments that the function requires. When you invoke the function, you pass to it variable names or values to substitute for these arguments.
To edit an existing argument, double-click its name. You can also edit an existing argument directly by clicking it twice or pressing F2.

To create a new argument, double-click the gray rectangle.

**Note:** Not all functions require arguments. You cannot use a function with arguments as an event handler.

**Opens the Function Parameter dialog box where you can define a new argument.**

**Deletes a selected argument from the list of arguments.**

**Moves a selected argument up in the list of arguments.**

**Moves a selected argument down in the list of arguments.**

**Returns: (Name/Type)**

Contains the name and data type of the return value that the function calculates. You can type this in, if you wish.

**Opens the Function Return dialog box, where you can define the return value.**

**Reference:**  **Function Editor: Description Tab**

When you click the Description tab in the New Function or Edit Function dialog box, WebFOCUS Maintain opens a dialog box where you can enter a description for your function, as shown in the following image.
You can view the description of a function in the Project Explorer with the Show description command. For more information, see *How to View the Description of a Procedure Component* on page 57.

**Example: Creating a Function**

Suppose you create a project that enables end users to update a list of fan club members. Part of this project is a form where end users can enter information for a new fan club member and then insert this information into the data source. This example demonstrates how you create a function that updates the data source and how you trigger it from the form.

1. In the Project Explorer, right-click the procedure you want the function to be part of.
2. In the shortcut menu, click New, then click Function (Case).
3. In the New Function dialog box, give your function the name *AddFan*, as shown in the following image.

![New Function dialog box](image)

4. Click OK.

Your new function appears in the list of components of the procedure.

5. Double-click *AddFan*.

WebFOCUS Maintain opens the source code for your function in the Procedure Editor.

6. Between Case *AddFan* and EndCase, enter the following Maintain language code:

```
For all include Fannames.CUSTOMER.SSN from AddFanStack;
Stack clear AddFanStack;
```

**Tip:** You can use the Language Wizard to enter this code instead of typing it. Place your insertion point between Case *AddFan* and EndCase, right-click in the Procedure Editor window, and in the shortcut menu, click *Language Wizard*.

7. Open the form from which you will execute this function.

8. Create a button named *AddButton* with the text *Add*, as shown in the following image.

![Add button](image)
9. Double-click the Add button to open the Events dialog box.
10. Select Click from the list of events.
11. Ensure you can see both the Events dialog box and the Project Explorer.
12. Click and drag the AddFan function from the Project Explorer into the Events dialog box.

   The Events dialog box looks similar to the following example:

   ![Events dialog box example]

13. Close the Events dialog box.

   When you execute the project, clicking Add executes the function AddFan.

**Using Variables in Procedures**

A variable is a named storage location capable of containing a certain type of data that can be modified during program execution. You also create variables as part of a class definition or function.

Maintain variables have four attributes:

- A name
- A data type (for example: integer, alphanumeric, or a user-defined class)
- A value (optional)
- A description (optional)
**Procedure:** How to Create a Variable in a Procedure

1. Select the procedure that you want the variable to be part of.

2. Right-click the procedure, click New in the shortcut menu, and click Variable (Declare) in the submenu.

   or

   Click the New variable button on the Maintain Objects toolbar.

3. In the New Variable dialog box, type a name for your variable in the Name box.

4. Specify a data type for your variable. You can do this in one of two ways:

   - Click the ellipsis button to open the Wizard. For more information, see Using the Type Wizard on page 78.

   or

   - The Type field contains a drop-down list of recently-applied data types. Select one from the list.

5. Optionally, specify an initial value for your variable by clicking the x=? Initialize tab and typing a value in the box.

6. Optionally, click the Description tab and type a description.

7. Click OK.

This variable is global, meaning it is available to any code written in this procedure. To use it in another procedure, you must pass it to the other procedure. For more information, see Passing Parameters Between Maintain Procedures: FROM...INTO on page 301.

**Procedure:** How to Edit a Variable

1. Right-click the variable in the Project Explorer.

2. In the shortcut menu, click Edit.

3. Make any necessary changes in the Edit Variable dialog box.

4. Click OK.

**Procedure:** How to Edit Source Codes in a Variable

WebFOCUS Maintain generates Maintain language code for variables. You can edit this code directly in the Procedure Editor.
1. Right-click the variable in the Project Explorer.

2. In the shortcut menu, click Edit Source.

3. Make your desired changes to the code after the DECLARE variablename keyword and before the semicolon (;).
   For more information, see the Maintain Language Reference manual.

4. Close the Procedure Editor.

   **Note:** You can obtain Help on any Maintain language keyword by selecting the keyword and pressing F1 key.

---

**Procedure:** How to Move a Variable to Another Procedure

1. In the Project Explorer, ensure that you can see both the variable and the procedure to where you want to move it.

2. Select the variable and drag it to the procedure to where you want to move it.

**Procedure:** How to Copy a Variable to Another Procedure

1. In the Project Explorer, ensure that you can see both the variable and the procedure to where you want to move it.

2. Select the variable, hold down the Ctrl key, and drag the variable to the procedure to where you want to move it.

**Reference:** Variable Editor: Declaration Tab

When you create or edit a variable, WebFOCUS Maintain opens the New Variable or Edit Variable dialog box.
When you define a function argument or return value from the Edit Function dialog box, define a variable as part of a class, or define a computed column for a data source stack, WebFOCUS Maintain opens the Function Parameter or the Function Return, or the Computed Stack Column dialog box. This dialog box looks like the Edit Variable dialog box, shown in the following image, but without the x=? Initialize tab.

This dialog box includes the following options:

**Name**
Contains the name of the variable, argument, or return value you are specifying.

**Type**
Contains the data type of the variable or argument or return value you are specifying. If you want to specify a data type, you can select a recently used data type from the list or type in the specification for the data type.

Opens the Type Wizard, where you can specify a new data type for the argument or return value. For more information, see *Using the Type Wizard* on page 78.

**Note:** The Type Wizard does not create specifications for the following data types: date-time, packed-decimal, and text. For more information on specifying packed-decimal or text, see the *Describing Data With WebFOCUS Language* manual.
**Reference: Variable Editor: \textit{x=? Initialize Tab}**

When you are creating or editing a variable, use the \textit{x=? Initialize} tab in the New Variable or Edit Variable dialog box to enter a value for your variable, as shown in the image below. This value can be any expression. For more information, see \textit{Expressions Reference} in the \textit{Maintain Language Reference} manual.

![Variable Editor: x=? Initialize Tab](image)

---

**Reference: Variable Editor: Description Tab**

When you are creating or editing a variable, function argument, function return value, or computed data source stack column, use the Description tab to enter a description, as shown in the image below.

You can view the description of a variable in the Project Explorer with the Show description command. For more information, see \textit{How to View the Description of a Procedure Component} on page 57.

![Variable Editor: Description Tab](image)
Using Data Source Stacks in Procedures

A data source stack is a non-persistent (or in-memory) array where you can store and manipulate data from one or more data sources. In the Maintain language, you use data source stacks to hold values you read from the data source and manipulate data before writing it back to the data source. You can think of a data source stack as a staging area in memory in which you can manipulate data before committing it to the data source or displaying it to the end user.

To use a data source stack, first create one using the Stack Editor, or by editing the procedure as text and typing the appropriate syntax. The columns of a stack are usually based on actual data source fields, but you can also add your own columns.

Fill the data source stack with data from one or more data sources. You can then use a form to display the stack data to the user. The user has the option of changing this data. WebFOCUS Maintain stores the changes in the data source stack until the project issues an UPDATE command to update the data source.

There are two ways to create a data source stack:

- Explicitly, by declaring the data source stack using an INFER statement in your procedure (use the Stack Editor to create the INFER statement or edit the procedure as text and type the INFER statement directly). For more information on using the Stack Editor, see *How to Create a Data Source Stack Explicitly Using the Stack Editor* on page 73.

- Implicitly, by loading data into the data source stack using the NEXT or MATCH commands.

For more information, see the *Maintain Language Reference* manual.

**Note:** When you use a data source field as a column in a data source stack, WebFOCUS Maintain defines columns based on the rest of the fields in that data source or, if your data source is hierarchical, the rest of the fields in the segment and the key fields in any parent segment. These columns are called implied columns. WebFOCUS Maintain includes all of these fields in your data source stack so that when you update your data source from the data source stack, it knows the path to these fields.
Procedure: How to Create a Data Source Stack Explicitly Using the Stack Editor

1. Ensure that all the data sources you want to reference in this stack are being used by the procedure. See Specifying Data Sources for Your Procedure on page 58.

2. Select the procedure that you want the data source stack to be part of.

3. Right-click the procedure, click New in the shortcut menu, and click Data Source stack in the submenu.

or

Click the New data source stack button on the Maintain Objects toolbar.

4. In the Stack Editor dialog box, enter a name for your data source stack.

5. Define the columns that make up the data source stack. There are two ways to do this (you can use one or both):

   - **Base the columns on actual fields in your data source.** Select a field under the list of data sources in the Available fields list and click the right arrow button.

   - **Create custom (user-defined) columns.** You can use custom columns as temporary work areas or to create calculated fields, such as Name = First || Last. Under the Variables folder, click New Variable. In the Computed Stack Columns dialog box under the Declaration tab, enter a name, and type for the new column. Then click OK. Repeat these steps until all of your columns have been defined.

6. If you wish, click the Description tab and type a description.

7. Click OK.

WebFOCUS Maintain creates an entry for your new data source stack in the Project Explorer under the procedure. When you open it, you see the columns that define the data source stack.
**Tip:** You can create a data source stack based on the fields in a data source segment by dragging the data source segment to the procedure in which you want to create the data source stack. WebFOCUS Maintain prompts you for the name of the data source stack in the Create New Stack dialog box. Creating a data source stack this way also uses the data source in the procedure, if you have not specified it already.

**Procedure:** How to Edit Source Codes in a Data Source Stack

WebFOCUS Maintain generates Maintain language code for data source stacks. You can edit this code directly in the Procedure Editor.

1. Right-click the data source stack in the Project Explorer.
2. In the shortcut menu, click *Edit Source*.
3. Make your desired changes to the code in the INFER statement.
4. Close the Procedure Editor.

**Note:** You can obtain Help on any Maintain language keyword by selecting the keyword and pressing F1 key.
**Reference: Stack Editor: Definition Tab**

When you create or edit a data source stack, WebFOCUS Maintain opens the Stack Editor dialog box, as shown in the following image.

This dialog box includes the following options:

**Available fields**

Lists the data sources that you have specified for use in this procedure. To define columns for the data source stack, either use existing fields from the data sources or create new ones by clicking *New variable*.

Copies a selected parameter or field into the list of data source stack columns.
Removes a selected column from the list of data source stack columns.

**Stack columns**

Contains the columns that you have defined for your data source stack.

When you use a data source field to define a stack, WebFOCUS Maintain defines columns for all of the rest of the fields in the data source. If your data source is hierarchical, WebFOCUS Maintain defines columns for all of the rest of the fields in the segment, and the key fields from parent segments. These are the implied columns.

**Infer datasource.segment.field into stack; and/or Compute stack.column/format;**

Contains the Maintain language code generated by the Stack Editor dialog box for your data source stack.

**Reference: Stack Editor: Description Tab**

When you are creating or editing a data source stack, use the Description tab in the Stack Editor dialog box to enter a description, as shown in the image below.
You can view the description of a data source stack in the Project Explorer using the Show description command. For more information, see *How to View the Description of a Procedure Component* on page 57.

**Reference:**  **Create New Stack Dialog Box**

Use the Create New Stack dialog box, shown in the following image, to name a data source stack that you have created by dragging a segment from a data source to a procedure.
Using the Type Wizard

This dialog box contains the following:

**New stack name**

Displays a proposed name for your new data source stack, which the name of the segment is, followed by Stk. You can accept this name or type a new one of 66 characters or less, with no spaces.

**Using the Type Wizard**

When you create a variable or define an argument or return variable, you must assign a data type to it. A data type is either a format or a class. Assign data types using the Type Wizard.

There are two kinds of data types: Built-in Type (or format) and User-defined Class.

If you choose Built-in Type, you have three basic formats:

- **Alphanumeric.** For values composed of alphabetic, numeric, or special characters.
- **Date.** For dates or date components.
- **Numeric.** For values composed of the digits 0 through 9 and optionally, a minus sign. You have three choices:
  - **Double Precision.** For whole numbers or fractions, using 8 bytes of storage and a maximum number of 15 positions. This type of field automatically inserts a comma after every third significant digit.
  - **Floating Point.** For whole numbers or fractions, using 4 bytes of storage and a maximum number of 9 positions.
  - **Integer.** For whole numbers, using 4 bytes of storage and a maximum number of 9 positions. Numbers after the decimal are truncated.

If you choose User-defined Class, you must have already defined a class. For more information, see Developing Classes and Objects on page 327.

**Note:** The Type Wizard does not create specifications for the following data types: date-time, packed-decimal, and text. You must manually enter the specification in the data type field. For more information on specifying packed-decimal or text, see the Describing Data With WebFOCUS Language manual.
An example of the Type Wizard is shown in the following image.

![Type Wizard](image)

**Procedure:** How to Apply Alphanumeric Format

1. In the Type Wizard, select *Built-in Type* from the list.
2. Select *Simple* or *Stack of* from the list.
3. Select *Alphanumeric* from the list of formats.
4. In the *of size* box, enter a length for your alphanumeric field (from 1 to 256 characters).
   
   or
   
   Select the *Variable length* check box to have a variable length field.

5. Click OK.

**Procedure:** How to Apply Date Format

1. In the Type Wizard, select *Built-in Type* from the list.
2. Select *Simple* or *Stack of* from the list.
3. Select *Date* from the list of formats.
4. Select a format from the list of date formats. This determines how your date appears when you view it.

5. Click OK.

**Procedure: How to Apply Numeric Format**

1. In the Type Wizard, select *Built-in Type* from the list.
2. Select *Simple* or *Stack of* from the list.
3. Select *Integer*, *Floating Point*, or *Double Precision* from the list of formats.
4. In the of size box, enter a width. This is the maximum number of positions in your field, comprising of all the digits (including any past the decimal point), the decimal point, any edit options, and if necessary, a minus sign.

For Double Precision and Floating Point formats, you must enter the number of places after the decimal point. The maximum number that you can enter depends on what numeric format you chose.

5. In the Display Options box, select any combination of the following:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suppress zeros</td>
<td>Suppresses leading zeros and displays a space if the value is 0 (zero).</td>
</tr>
<tr>
<td>Leading zeros</td>
<td>Adds leading zeros to the full field length.</td>
</tr>
<tr>
<td>Include commas</td>
<td>Inserts a comma after every third significant digit. This option takes effect automatically for Double Precision format.</td>
</tr>
<tr>
<td>Floating dollar</td>
<td>Adds a dollar sign immediately to the left of the number and inserts a comma after every third significant digit.</td>
</tr>
<tr>
<td>Fixed dollar</td>
<td>Adds a dollar sign to the left of the field and inserts a comma after every third significant digit.</td>
</tr>
<tr>
<td>Scientific notation</td>
<td>Displays the number in scientific notation.</td>
</tr>
<tr>
<td>Credit negatives</td>
<td>Adds the characters CR after a negative number.</td>
</tr>
<tr>
<td>Bracket negatives</td>
<td>Encloses a negative number in parentheses.</td>
</tr>
</tbody>
</table>
6. Click OK.

**Procedure: How to Apply a User-defined Class**

1. In the Type Wizard, select *User-defined Class* from the drop-down list.

   **Note:** User-defined classes are available only if you defined classes in the procedure.

2. Select *Simple* or *Stack of* from the drop-down list.
3. Select a class from the list.
4. Click OK.

   For more information on defining classes, see *Developing Classes and Objects* on page 327.

**Using Import Modules**

An import module is a non-executable procedure that contains Maintain language code. You do not execute an import module as part of your project. Instead, you use it as a library of functions or classes. Using the `MODULE IMPORT` command, you can bring an import module into any procedure and use its functions and classes. An import module cannot be used for data source access.

If you edit an import module, you will notice that, just as with a procedure, it starts with the keyword `MAINTAIN` and ends with the keyword `END`. However, it does not contain a Top function.

You can use the following in an import module:

- Functions
- Classes
- Variables

You cannot use the following in an import module:

- Forms.
- Data sources (the syntax `MAINTAIN FILE filename` is not supported).
- Commands that read or write data to a data source (NEXT, MATCH, and so on).
- Stacks that contain columns from a data source (defined with the INFER or NEXT commands). You can, however, define stacks using Stack Of, DECLARE, or COMPUTE.
**Note:** When you move an import module into a procedure, the variables, functions, and classes in an import module become part of the name space of the procedure. For example, if you have a variable named COUNT in an import module and import it into a procedure, that procedure cannot define its own variable named COUNT.

**Procedure:** How to Use an Import Module in a Procedure

1. Right-click the procedure.
2. In the shortcut menu, click *Import modules*.
   
The Use these Data Sources in Procedure dialog box opens.
3. Select the import modules you want to use in the Available import modules list and click the right arrow button. 
4. To remove import modules you no longer need, select them in the Modules to import list and click the left arrow button. 
5. Use the *Move down* and *Move up* buttons to change the order of the import modules (the order matters only if import modules have identically named functions).
6. Click *OK*.

WebFOCUS Maintain updates the source text of the importing procedure. If this is the first time you specify an import module for the procedure, WebFOCUS Maintain creates an Import Modules folder under the procedure and lists in it the import modules you specified.

**Tip:** You can also drag an import module to the procedure in which you want to use it.
**Reference:** Import These Modules Into Procedure Dialog Box

Specify the import modules to use in your procedure with the Import these Modules into Procedure dialog box, shown in the following image.

![Import these Modules into Start dialog box](image)

This dialog box includes the following options:

**Available import modules**

Contains a list of the import modules that are part of your project.

Copies a selected import module into the list of modules to import.

Removes a selected import module from the list of modules to import.

**Modules to import**

Contains a list of the import modules this procedure uses.

Moves a selected import module up in the list of import modules.
Moves a selected import module down in the list of import modules.

Testing Procedures With the Run Procedure Option

You can test a procedure by conducting a local deploy and run using the Run Procedure option.

Procedure: How to Test Procedures With the Run Procedure Option

1. Right-click a procedure in the Object Explorer or Project Explorer window.
2. Select Run Procedure.

This option causes a local deploy and run.
Using the Form Editor

The Form Editor enables you to design the user interface for your application. You can use controls such as list boxes, buttons, and input fields to make your application easy for your users to navigate.

These topics provide information on how to use the Controls palette to place controls on your interface, how to use the property sheet to define application properties and define events in your application, and how to use drawing aids to improve the appearance of your application.

In this chapter:

- Form Editor Overview
- Layout of the Form Editor
- Using the Controls Palette
- Using the Property Sheet
- Using Drawing Aids
- Selecting Multiple Controls
- Cutting, Copying, Pasting, and Duplicating Controls
- Resizing Controls
- Aligning Controls
- Spacing Controls
- Grouping Controls
- Changing the Order of Controls
- Undoing and Redoing Actions
- Layering Controls
Form Editor Overview

The Form Editor designs the user interface for your application. End users interact with the application you design through the use of controls like list boxes, buttons, and input fields. You can also style your forms with frames, text, lines, and images. In addition, you can create forms that run reports, search data sources, and much more.

Here is a brief overview of how you design forms:

1. Place controls, such as radio buttons, check boxes, images, and buttons, on your form to communicate information to and collect information from your end users.

   For example, you use edit boxes, list boxes, check boxes, and radio buttons to request information from end users. You use text and images to communicate information to them.

   For more information on how to use controls, see Developing and Using Controls on page 163.

2. Define the properties of these controls. The properties determine what the controls look like (for example, colors and fonts) and how they behave (for example, whether an entry into an edit box displays asterisks for passwords, or whether the end user can make multiple selections in a list box). Some controls also have properties which can be bound to data, either from a Maintain variable or coded at development time.

   Most properties can be changed dynamically at run time.

   For more information on properties, see Form and Control Properties Reference on page 267.

3. Define the events that end users cause by interacting with the form (such as clicking a button) and what action occurs when an event takes place (such as writing data to a data source or opening another form).

   For more information on events, see Defining Events and Event Handlers on page 125.

After you place controls on your form, you can:

- Select all the controls on your form and manipulate them as a group.
- Use standard Cut, Copy, and Paste commands on your controls. WebFOCUS Maintain also supplies an extra command for making copies of controls, the Duplicate command.
- Use special commands to resize, align, and space controls.
- Group controls together so that they are treated as a unit.
- Undo editing mistakes or redo actions.

If you are designing complicated forms, you can place controls on separate layers.
Layout of the Form Editor

The Form Editor design tools are identified in the following image.

The following are the main components of the Form Editor:

**Form**
Is your main working area. What you place here is what appears on the form end users see.

**Controls palette**
Contains the controls you can place on your form, such as text, radio buttons, check boxes, and so on.

**Property sheet**
Displays a list of properties for the form or selected control. Properties determine what the form and controls look like and how they behave at run time.

**Rulers**
Helps your position controls on a form by displaying inch markings at the top and left of the form.
Guidelines

Determines a border area around the outside edge of your form where you cannot place controls.

Layout toolbar

Contains commands that manipulate forms and controls on a form.

Notice also that the menu bar has a new menu, the Layout menu.

Using the Controls Palette

The Controls palette, shown in the image below, contains the controls that you can place on your form.

![Controls Palette](image)

The first control, which is the Select control, is the only one that does not place something on your form. Click this control if you want to be able to select controls on your form. This is the default control selected.

Procedure: How to Place a Control on Your Form

1. Click the control in the Controls palette.
2. Move your cursor to the form and notice that your cursor is now a cross-hair. Draw a rectangle in the approximate location at the approximate size you want your control.

   You can adjust the size and location of your new control later.

For some controls, the Form Editor immediately opens a dialog box that you must fill out to specify some information about the control (for example, if you place an image on your form, the Form Editor immediately opens the Image Source dialog box).

For specific information about each control, see Developing and Using Controls on page 163.
Using the Property Sheet

The property sheet enables you to view and edit properties and events for your forms and controls.

At the top of the property sheet is an alphabetical list of the controls in your form. You can display the properties of a control by selecting its name from the list.

If you select a group of controls, the list contains the individual controls in that group. You can easily change the properties of one of these controls by selecting it from the list.

The Properties tab in the property sheet contains an alphabetical list of properties for the currently selected control. If no control is selected, then the property sheet contains the properties for the current form.

The properties determine what your forms and controls look like and how they behave at run time. Some properties can be bound to data. These properties show up in blue in the property sheet.

You can also set many properties dynamically, meaning you can insert code in your procedures to change the values of these properties at run time. For example, you might want to display some text to end users after they click a button. To do this, place the text on the form and set its Visible property to No. When end users click the button, you would dynamically change the Visible property for the text to Yes, thus displaying the text.

You can type the syntax into the procedure to change these properties, or you can drag the property into the procedure, and WebFOCUS Maintain generates the correct syntax for you. For more information on this syntax, see How to Set the Value of a Form Property on page 120, How to Set the Value of a Control Property on page 259, and How to Set the Value of a Control Color Property on page 262.

**Note:** You cannot drag properties that cannot be set at run time.
An example of the property sheet is displayed in the following image. For a description of each property, see *Form and Control Properties Reference* on page 267.

The Events tab in the property sheet contains a list of events for the selected control or form. Events are changes in the state of a form or control that generates an action, called an event handler. You can see which events have event handlers in the Events tab. You can also open the Event Handler editor by double-clicking an event.

An example of the Events tab is displayed in the following image. For more information on events, see *Defining Events and Event Handlers* on page 125.
Procedure: How to Change a Property

1. On the form, select the control whose property you want to change. If you want to select the form, click anywhere in the form outside of a control or, in the property sheet, select the control from the list of controls at the top.

2. Find the property you want to change (you may need to scroll down in the list to find the one you want).

3. For some properties, you can enter the value directly in the box to the right of the property name (for example, the (Name) property), as shown in the following image.

   ![Property Editor](image)

   For some properties, you must select the value from a drop-down list, shown in the following image. Click the List button to the right of the property to see a list of values and select one.

   ![Drop-down List](image)

   For some properties, you must open a dialog box to determine the value for the property. Double-click the property or select the property and click the ellipsis button in the box to the right of the property.

Procedure: How to Set Properties Dynamically

Instead of manually coding the syntax to set the property, you can do the following:

1. To set a property in the Events window, open the Events window. To set a property in a procedure, open the procedure in the Procedure Editor. Make sure that you can see the window and the property sheet.

2. Select the control whose property you want to set dynamically. If you want to select the form, click anywhere in the form outside of a control or, in the property sheet, select the control from the list of controls at the top.

3. Select the property you want to set dynamically by clicking it. You may need to scroll down in the list to find the one you want. You cannot dynamically set bindable properties, which are the properties in blue.

4. Click the property again and drag it to the window where you want to insert this statement. If you cannot drag the property, which means that you cannot set it dynamically.
5. By default, the syntax generated is the current value of this property. If you wish, change the value for the property setting. *Form and Control Properties Reference* on page 267 contains a complete list of settings for each property.

**Using Drawing Aids**

The Form Editor provides you with the following drawing aids to help you position controls on a form:

- You can display a drawing grid to help you position controls on a form. A drawing grid contains a series of dots that represent where the horizontal and vertical lines intersect. You can force your controls to snap to these lines, or you can just use the grid as a visual aid to help position and align controls.

- You can use guidelines at the edges of your form to ensure that controls are not placed within a certain distance from the border of the form.

- You can display rulers at the top and left sides of your form.

You can also align controls to each other. For more information, see *Aligning Controls* on page 98.

**Procedure: How to Display a Drawing Grid and Guidelines**

To display a drawing grid and guidelines, do one of the following:

- In the View menu, click *Grid*.

- Click the *Toggle grid* button on the Layout toolbar.

**Procedure: How to Change Drawing Grid Settings**

1. In the Layout menu, click *Grid settings*.
   
   The Grid Settings dialog box opens.

2. Type the size you want for each grid block in the Guidelines box.

3. Type the width and height of the grid in the Spacing box.
   
   The width and height of the grid will be measured against the edges of the form.

4. Select one or all of the grid setting options:
   
   - *Show Grid* displays a grid on a form.
   
   - *Snap to Grid* automatically snaps controls to the closest gridline.
Snap to Center automatically centers each control around the gridline closest to the center of the control, in relation to its borders. In this mode, resizing of controls is prevented. Snap to Grid must be enabled before selecting the Snap to Center option.

5. If you want these settings to apply to this form only, select Apply to this form only.

Procedure: How to Display Rulers

1. Make sure that the form you want to display rulers in is the active window.
2. In the View menu, click Rulers.
   or
   Click the Toggle rulers button on the Layout toolbar.

Reference: Grid Settings Dialog Box

You use the Grid Settings dialog box, shown in the following image, to determine settings for your grid.

Show Grid
Displays the grid on the form.

Snap to Grid
Causes controls to snap to the gridlines.
Selecting Multiple Controls

To save time, you can select multiple controls and perform many operations on the entire group. The Form Editor identifies the first control you select as the anchor (reference) control, and all the others as targets. The attributes of the anchor determine the attributes of the targets when you align or position the controls.

The anchor control has green handles, and other selected controls have blue handles.

Procedure: How to Select Multiple Controls

To multi-select a group of controls, do one of the following:

- Hold down the Shift key while selecting each object.
- Draw a rectangle around the controls by dragging, and then releasing the left mouse button.

Note: To select all the controls on a form, in the Edit menu, click Select all, or press Ctrl+A.

Cutting, Copying, Pasting, and Duplicating Controls

If you want to place a control on a form that is very similar to an existing control, instead of generating a new control, you can use the Cut, Copy, Paste, and Duplicate commands.

When you copy and paste a control, you can perform either a deep copy or a shallow copy:

- A deep copy copies the appearance of the control (for example, the button and its text) as well as the data bindings, the connections to resources, and the event handlers (that is, its behavior).
You can perform deep copies using the Paste and Duplicate commands.

- A shallow copy copies appearance only.

You can perform shallow copies using the Paste appearance command.

**Procedure: How to Cut a Control**

1. Select the control.
2. In the Edit menu, click *Cut*.
   - or
   - Press Ctrl+X.
   - or
   - Click the *Cut* button on the General toolbar.
   - or
   - Right-click the control, and in the shortcut menu, click *Cut*.

WebFOCUS Maintain removes the control from the form and places it on the Clipboard, ready to paste into this or another form.

**Procedure: How to Copy a Control**

1. Select the control.
2. In the Edit menu, click *Copy*.
   - or
   - Press Ctrl+C.
   - or
   - Click the *Copy* button on the General toolbar.
   - or
   - Right-click the control, and in the shortcut menu, click *Copy*.

WebFOCUS Maintain places the control on the Clipboard, ready to paste into this or another form.
**Procedure: How to Paste a Control**

Before you paste a control into a form, you must have cut or copied it to the Clipboard.

If you want to perform a deep copy (includes appearance, data bindings, event handlers, and connections to resources):

1. Open the form where you want to paste the control.
2. In the Edit menu, click *Paste*.
   
   or

   Press Ctrl+V.

   or

   Click the *Paste* button on the General toolbar.

   or

   Right-click the form, and in the shortcut menu, click *Paste*.

WebFOCUS Maintain places the control in the form at its previous location or, slightly offset from its previous location. It also attempts to do the deepest possible copy, but a full copy may not be possible. For example, you cannot perform a deep copy from a form in one procedure to a form in another procedure.

If you want to perform a shallow copy (appearance only):

1. Open the form where you want to paste the control.
2. In the Edit menu, click *Paste appearance*.

   or

   Right-click the form, and in the shortcut menu, click *Paste appearance*.

**Procedure: How to Duplicate a Control**

1. Select the control.
2. In the Edit or Form menu, click *Duplicate*.

   or

   Right-click the control and in the context menu, click *Duplicate*.

WebFOCUS Maintain creates a deep copy of the control (including appearance, data bindings, events, and connections to resources) in the same form, slightly offset from the original control.
Resizing Controls

You can easily resize a control by selecting it and dragging its handles. However, you may often want one control to be the same size, height, or width as another control on the form. WebFOCUS Maintain provides you with several options for automatically resizing a control.

**Procedure: How to Make Controls the Same Width**

1. Select the control whose width you want other controls to use. This selected control is the anchor control. Notice that the handles on this control are green.
2. Select the controls whose width you want to change. Notice that the handles on these controls are blue.
3. In the Layout menu, click *Make same size*, and then click *Width* in the submenu. 
   or
   Click the *Make Same Size* button in the Layout toolbar. Then click the *Width* button in the pop-up toolbar.

**Procedure: How to Make Controls the Same Height**

1. Select the control whose height you want other controls to use. This selected control is the anchor control. Notice that the handles on this control are green.
2. Select the controls whose height you want to change. Notice that the handles on these controls are blue.
3. In the Layout menu, click *Make same size*, and then click *Height* in the submenu.
   or
   Click the *Make Same Size* button in the Layout toolbar. Then click the *Height* button in the pop-up toolbar.

**Procedure: How to Make Controls the Same Size**

1. Select the control whose size you want other controls to use. This selected control is the anchor control. Notice that the handles on this control are green.
2. Select the controls whose size you want to change. Notice that the handles on these controls are blue.
Aligning Controls

3. In the Layout menu, click Make same size, and then click Both in the submenu.
   
or

   Click the Make Same Size button [ ] in the Layout toolbar. Then click the Size button [ ] in the pop-up toolbar.

Aligning Controls

Your forms will look more polished if the controls are neatly aligned. WebFOCUS Maintain provides many alignment tools to help align your controls.

You can align controls:

- By their left edges.
- By their right edges.
- By their tops.
- By their bottoms.
- By their centers (either horizontally or vertically).

You can align prompted edit boxes along the left sides of the edit boxes.

Procedure: How to Align Controls Along Their Left Sides

To align controls along their left sides:

1. Select the control to whose left side you want the other controls to be aligned. This is the anchor control. Notice that the handles on this control are green.
2. Then select the controls you want to move. Notice that the handles on these controls are blue.
3. In the Layout menu, click Align, and then click Left in the submenu.
   
or

   Click the Align Edges button [ ] in the Layout toolbar. Then click the Left button [ ] in the pop-up toolbar.

Procedure: How to Align Controls Along Their Right Sides

To align controls along their right sides:
1. Select the control to whose right side you want the other controls to be aligned. This is the anchor control. Notice that the handles on this control are green.

2. Then select the controls you want to move. Notice that the handles on these controls are blue.

3. In the Layout menu, click Align, and then click Right in the submenu.

   or

   Click the Align Edges button in the Layout toolbar. Then click the Right button in the pop-up toolbar.

**Procedure: How to Align Controls Along Their Tops**

To align controls along their tops:

1. Select the control to whose top you want the other controls to be aligned. This is the anchor control. Notice that the handles on this control are green.

2. Then select the controls you want to move. Notice that the handles on these controls are blue.

3. In the Layout menu, click Align, and then click Top in the submenu.

   or

   Click the Align Edges button in the Layout toolbar. Then click the Top button in the pop-up toolbar.

**Procedure: How to Align Controls Along Their Bottoms**

To align controls along their bottoms:

1. Select the control to whose bottom you want the other controls to be aligned. This is the anchor control. Notice that the handles on this control are green.

2. Then select the controls you want to move. Notice that the handles on these controls are blue.
3. In the Layout menu, click Align, and then click Bottom in the submenu.

or

Click the Align Edges button in the Layout toolbar. Then click the Bottom button in the pop-up toolbar.

Procedure: How to Align Controls Along Their Centers

To align controls along their centers:

1. Select the control to whose centers you want the other controls to be aligned. This is the anchor control. Notice that the handles on this control are green.

2. Then select the controls you want to move. Notice that the handles on these controls are blue.

3. In the Layout menu, click Align, and then click Center horizontally or Center vertically in the submenu.

or

Click the Align Edges button in the Layout toolbar. Then click one of the Center buttons in the pop-up toolbar.

Procedure: How to Align Prompted Edit Boxes Along the Left Sides of the Edit Boxes

1. Select the control to whose centers you want the other controls to be aligned. This is the anchor control. Notice that the handles on this control are green.

2. Then select the controls you want to move. Notice that the handles on these controls are blue.

3. In the Layout menu, click Align, and then click Center vertically in the submenu.

or

Click the Align Edges button in the Layout toolbar. Then click the Center vertically button in the pop-up toolbar.
Spacing Controls

The WebFOCUS Maintain spacing commands determine the two outer controls in a set of controls that you have selected, and then evenly space the rest of the controls between the outer controls. The spacing commands can save you a lot of time while designing forms.

**Procedure: How to Space Controls**

1. Select the controls you want to space. You must select at least three controls.
2. In the Layout menu, click Space, and then click Horizontally or Vertically in the submenu.
   
   or

   Click the Space button in the Layout toolbar. Then click one of the Space buttons in the pop-up toolbar.

Grouping Controls

You can use the WebFOCUS Maintain grouping commands to treat several controls as a unit. This means that when you move one of these controls, WebFOCUS Maintain moves all of them together. You can also apply alignment and spacing commands to these controls, and WebFOCUS Maintain treats them as one control.

When you select a set of grouped controls, the name of the group appears in the property sheet in a combo box. You can change the properties for an individual control in the group by selecting the control from the drop-down list.

**Procedure: How to Group Controls**

1. Select the controls you want to group.
2. In the Layout menu, click Group.
   
   or

   Click the Group controls button in the Layout toolbar.

**Procedure: How to Ungroup Controls**

1. Select the controls you want to ungroup.
2. In the Layout menu, click Ungroup.
Changing the Order of Controls

or

Click the Ungroup controls button in the Layout toolbar.

Procedure: How to Regroup Controls

- In the Layout menu, click Regroup.

- Click the Regroup controls button in the Layout toolbar.

Changing the Order of Controls

If two controls occupy the same space on your form, by default, the control that was created most recently appears on top. If you want to change the order of controls on your form, use the Bring to Front or Send to Back commands.

The order of controls in the form may affect the tab order.

Note: WebFOCUS Maintain will not move a button, an edit box, or a multi-edit box behind a group box, static text, line, Java applet, HTML Object, or HTML Table, since doing so would make it inaccessible to end users. This is only applicable to Bring to Front and Send to Back commands. It does not apply to controls being manipulated by the ZIndex property.

Procedure: How to Send a Control to the Front

1. Select the control.

2. In the Layout menu, click Bring to front.

   or

   Click the To Front or Back button in the Layout toolbar. Then click the To front button in the pop-up toolbar.

Procedure: How to Send a Control to the Back

1. Select the control.

2. In the Layout menu, click Send to back.

   or
Click the To Front or Back button in the Layout toolbar. Then click the To back button in the pop-up toolbar.

**Note:** Another option is the ZIndex property, which can be used to assign display order for two or more controls on a form, either during development or dynamically at run time (for all controls except Grid and ActiveX). For more information on this property, see [ZIndex Property](#) on page 294.

### Undoing and Redoing Actions

You can undo or redo an action in the Form Editor.

**Procedure:** How to Undo an Action

To undo an action in the Form Editor, do one of the following:

- In the Edit menu, click *Undo*.
- Right-click the form, and in the shortcut menu, click *Undo*.
- Press Ctrl+Z.
- Click the *Undo* button in the General toolbar.

**Procedure:** How to Redo an Action

To redo an action in the Form Editor, do one of the following:

- In the Edit menu, click *Redo*.
- Right-click the form, and in the shortcut menu, click *Redo*.
- Press Ctrl+Y.
- Click the *Redo* button in the General toolbar.

### Layering Controls

By default, all the controls on a form are placed on the same layer (the Default layer). If you are designing complicated forms with many controls, you can take advantage of the layering ability of the Form Editor.
At design time, you can create multiple layers and then assign different controls on the form to different layers. You can then hide a layer, which hides all the controls on that layer until you make it visible again. You can also lock a layer, which ensures that all controls on the layer cannot be changed. For run time, layers can be dynamically displayed or hidden using the SetLayer command. An example of the Layers Sheet dialog box is shown in the image below.

![Layers Sheet dialog box](image.png)

**Procedure: How to Create a Layer**

1. In the Form menu, click *Edit layers*, or click the *Edit layers* button on the Layout toolbar to open the Layers Sheet dialog box.

2. Click *Add*.

3. If you wish, rename your layer by typing over the name in the Name box.

4. Click *OK*.

   The new layer is added on top of the existing layers.
**Procedure: How to Make a Layer Active**

To make a layer active so that all new controls will be placed on this layer:

1. In the Form menu, click *Edit layers*, or click the *Edit layers* button on the Layout toolbar to open the Layers Sheet dialog box.
2. Select the layer you want to make active.
3. Click *Select*.
4. Click *OK*.

**Procedure: How to Hide or Unhide a Layer**

1. In the Form menu, click *Edit layers*, or click the *Edit layers* button on the Layout toolbar to open the Layers Sheet dialog box.
2. Select the layer you want to hide or unhide.
3. Select or clear the Hide box.
4. Click *OK*.

**Procedure: How to Lock or Unlock a Layer**

1. In the Form menu, click *Edit layers*, or click the *Edit layers* button on the Layout toolbar to open the Layers Sheet dialog box.
2. Select the layer you want to lock or unlock.
3. Select or clear the Lock box.
4. Click *OK*.

**Procedure: How to Change the Order of the Layers**

1. In the Form menu, click *Edit layers*, or click the *Edit layers* button on the Layout toolbar to open the Layers Sheet dialog box.
2. Select the layer you want to move.
3. Click the *Up* or *Down* button.
4. Click *OK*.
**Procedure:** How to Move an Existing Control to Another Layer

1. Select the control.
2. In the property sheet, select the Layers property.
3. Select the name of the layer you want the control to be on.

**Procedure:** How to Rename a Layer

1. In the Form menu, click Edit layers, or click the Edit layers button on the Layout toolbar to open the Layers Sheet dialog box.
2. Select the layer you want to rename.
3. Type a new name in the Name box.
4. Click OK.

**Procedure:** How to Delete a Layer

1. In the Form menu, click Edit layers, or click the Edit layers button on the Layout toolbar to open the Layers Sheet dialog box.
2. Select the layer you want to delete.
3. Click Delete.
4. Click OK.

**Syntax:** How to Control Layers at Run Time Using the SetLayer Command

The SetLayer command allows layers to be set as visible or invisible at run time. The command can be used in a Maintain or JavaScript event. The syntax is

**Maintain**

`Formname.SetLayer("layername",n);`

**JavaScript**

`setLayer.formname.("layername",n);`

where:

"layername"

Is the name of the layer, enclosed in double quotation marks (".")
Possible values are:

- 0 sets the layer to invisible (off).
- 1 sets the layer to be visible (on).

To preset a layer before the form is run, you can use the SetLayer command with the Winform Show_inactive command:

```plaintext
Wiform Show_inactive Form1;
Form1.SetLayer("Layer1",1);
Form1.SetLayer("Layer2",0);
Wiform Show Form1;
```

This sets Layer1 as visible and Layer2 as invisible.
Layering Controls
A form is an interface that enables you to interact with an application. You can create an application using multiple forms.

This topic describes how to design, manage, and manipulate forms.

**In this chapter:**
- Before You Begin Designing Forms
- Creating and Managing Forms
- Copying a Form Within a Procedure
- Changing Form Properties
- Using Cascading Style Sheets
- Dynamically Manipulating Forms at Run Time
- How WebFOCUS Maintain Saves Your Forms
- Using a Driver Procedure

---

**Before You Begin Designing Forms**

Here are some points to keep in mind before you start your form development process:

- Try to adopt the point of view of the end user. You may view your application as a series of functions, but the end user sees the application as the interface.

- Consider developing a template for your forms. A template ensures that all of your forms look alike, and makes it easier for end users to navigate through your application. For example, if many of your forms have a Next button, putting it in the same place for every form ensures that end users become familiar with the application more quickly.

- Give your application the same look and feel as other applications or webpages at your site. Once again, this helps end users learn your application more easily and gives your application more credibility.
Determine the flow of all of the forms in your application and implement a driver procedure. Determine which forms you have, and which forms call each other. Then implement the driver procedure discussed in *Using a Driver Procedure* on page 122. Using the driver procedure reduces the likelihood of seeing one of the most common WebFOCUS Maintain application errors, the Stack Overflow error.

If any of your forms are reusable, place them in their own separate procedures and use the CALL command to execute them. For example, perhaps you have a Message Form for help messages, or a Browse Form for data browsing, or a Credit Card Form for secured credit card entry.

You can write your own customized JavaScript functions to be used every time a Maintain form is loaded or refreshed by including them in the mntonload.js file. For more information, see *Customizing Key Functions and JavaScript for WebFOCUS Maintain Applications* on page 348.

**Creating and Managing Forms**

You create and manage forms using the Project Explorer.

By default, when you create your Maintain procedure, WebFOCUS Maintain creates an initial form, named Form1, with the necessary code in the Top function to launch the form. You can rename this form. The new name is automatically reflected in the Winform Show statement.

**Note:**

- If you delete or rename a form, be sure to fix or comment out the Winform Show command in the Maintain procedure that calls it. If you skip this step, Project Explorer shows a marker in place of a deleted form. If you click on that marker, a message appears stating that the form does not exist and asking if you want to create it.

- After you delete the Winform Show command from a form, be sure to remove it or comment it out. If you skip this step, Project Explorer will show a marker in place of a deleted form. If you click on that marker, a message appears stating that the form does not exist and asking if you want to create it.

**Procedure:** How to Create a Form

1. Select the Maintain procedure that you want the form to be part of.

**Note:** You must deploy all forms in an application to the same WebFOCUS Server. Keep this in mind when deciding in which procedure to place your forms.
2. Right-click the procedure, click New in the shortcut menu, and click Form in the submenu. The new form opens in the Form Editor.
After you have created a form, it is displayed under its procedure in the Project Explorer.

**Procedure: How to Edit a Form**

1. In the Project Explorer, open the procedure that contains the form you want to edit.
2. Open the Forms folder.
3. Right-click the form, and in the shortcut menu, click Open.
   or
   Double-click the form.

The form opens in the Form Editor.

**Procedure: How to Rename a Form**

1. In the Project Explorer, right-click the form, and in the shortcut menu, click Rename.
   or
   Select the form and press F2.
   or
   Select the form, then click it again to edit its name.
2. Type your new name.
3. Press Enter to confirm your change.

**Exporting and Importing Forms**

If you want to base forms from different procedures on a single template, the Maintain Development Environment enables you to export a form to a file, and then import it into another procedure.

Keep in mind that when you import a form into a different procedure from the procedure you exported it from, WebFOCUS Maintain preserves bindings to resources such as web links, scripts, and graphics, but not to variables or data source stacks (since variables and data source stacks are defined only in the context of a procedure).
Procedure: How to Export a Form

1. In the Project Editor, right-click the form and click Export.
2. In the Export As dialog box, enter a name for your form and click Save.

WebFOCUS Maintain saves your form in a file with the extension .for in the project folder.

Procedure: How to Import a Form

1. In the Project Editor, right-click the Maintain procedure you want to import the form into and click Import forms.
2. In the Import Form dialog box, select the form you want to import and click Open.
   If you cannot see the form you want to import, make sure that it is in one of your project paths.

Copying a Form Within a Procedure

From within the Maintain Development Environment, you can copy a form within a procedure.

Procedure: How to Copy a Form Within a Procedure

To copy a form:

1. In the Maintain Development Environment Project Explorer, right-click a form name and select Copy form.
   Bindings to data source stacks and variables are preserved when you copy forms within a procedure. The new form will have the original form name with _copy1 (_copy2, and so on) appended to it.
2. It is recommended that you rename the new form to something appropriate for your application.

Changing Form Properties

When you click the background of a form (thus deselecting all of the controls in your form), you will see a list of form properties in the property sheet. Changing these properties changes how your form appears and how it behaves at run time.

If you want to change the title of the form that is displayed to the end user, use the Title Property on page 293.
If you want to change the color of the form, use the BackColor Property on page 269 or use a style sheet. For more information on style sheets, see Using Cascading Style Sheets on page 113.

If you want to add an image to the background of the form, use the BackgroundImage Property on page 270 or use a style sheet. For more information on style sheets, see Using Cascading Style Sheets on page 113.

If you want to change the font of all controls in the form, use the Font Property on page 279 to change the typeface, style, and size of the text, the ForeColor Property on page 279 to change the color of the text, or use a style sheet.

If you want to change what the cursor looks like when it is on top of the form, use the CursorPointer Property on page 276.

If you want to display a tooltip when the cursor is on the top of the form, use the ToolTipText Property on page 293.

If you want to assign a Help topic to the form, use the Help Property. For more information, see Assigning Help to Your Forms and Controls on page 263.

Using Cascading Style Sheets

You can use Cascading Style Sheets while developing WebFOCUS Maintain forms. Cascading Style Sheets enable you to format the appearance of your forms using one separate file, rather than having the formatting information appear through all of your files.

The formatting information for a form or control comes from one of three places, in the following order of precedence:

1. Set its value explicitly in the property sheet. This value takes precedence over any other setting.
2. Apply a style sheet to your form and apply a class to any control in your form. The control will use the characteristics of the class, as defined in the style sheet.
3. Apply a style sheet to your form (but do not apply any classes to the controls). The form and controls will use the default characteristics defined in the style sheet.

You can use all of these types of styling in the same Maintain form, allowing for powerful and flexible styling capabilities.

Procedure: How to Set a Style Sheet Using the Properties Sheet

You can apply a style sheet using the CSSName property in the Properties sheet of the form.

1. Click on the form to display its properties in the property sheet.
2. Enter the name of the style sheet as the value for the CSSName property. Enter the .css extension as part of the value.

**Note:** The referenced style sheet must be included in your project.

**Syntax:** How to Apply a Style Sheet to Your Form

To apply a style sheet to your form using Maintain code, place the following in your procedure:

```
Winform Show_Inactive form
Winform Set form.CSSname TO stylesheet;
Winform Show form
```

The form and all the controls in the form that have their properties set to Default, except for the Grids, use the default settings from the style sheet.

**Syntax:** How to Apply a Class to a Control on Your Form

Assuming you have already applied a style sheet to your form, you can apply a class to any control on your form using the following syntax:

```
Winform Set form.control.CSSCLASS TO classname;
```

The control inherits the characteristics of the class as specified in the style sheet. This means that all of the properties of that control that are set to Default use the setting for that class in the style sheet.

**Dynamically Manipulating Forms at Run Time**

Using the Winform command in the Maintain language, you can open, close, reset, and perform other operations on your forms at run time.

If you are developing with forms, you must understand the flow of control for forms in a WebFOCUS Maintain application. (Keep in mind that Maintain procedures do not execute linearly. The only code that is guaranteed to be executed is in and before the Top function.)

**Displaying Forms at Run Time**

The Maintain language has the following commands for displaying forms:

- The Winform Show_And_Exit form command displays a form, then severs the connection of the form to the server while continuing to display the form.

  This is non-persistent mode. It is the most common mode for Maintain forms. When an end-user submits a non-persistent form, the form re-establishes a connection with the server.
When you display a form with Winform Show_And_Exit, control passes to the form, and any commands after Winform Show_And_Exit are not executed until the form is closed.

- The Winform Show form command (or Winform Show_Active form) displays a form and makes it active so that an end user can use it.

When you display a form with Winform Show, control passes to the form, and any commands after Winform Show are not executed until the form is closed.

- The Winform Show_Inactive form command displays a form without making it active. Because the form is inactive, control passes to the next command. Use this command if you want to change some form properties dynamically before displaying the form.

To display an initial form, somewhere in the Top Case of the starting procedure, you must issue the Winform Show command that displays this form (or Top Case should call another function or procedure that executes this command).

The rest of the forms in the procedure are displayed only if the end user does something to display the forms, such as clicking a button. Clicking a button is known as an event to a WebFOCUS Maintain developer, and you use the Event Handler editor to define what happens when an event occurs. For more information, see Defining Events and Event Handlers on page 125.

As each form is opened in a WebFOCUS Maintain application, it is stacked on top of the previous form. This means that all your forms remain open until you issue a command to close them or exit the application.

**Note:** When you close a form, you close every form in its chain, and WebFOCUS Maintain ignores any code after the closing statement.

When WebFOCUS Maintain encounters the Winform Show[Active] command, it passes control to the named form. Control does not return to the procedure until the user exits the form.

**Syntax:** How to Make a Form Active

To make a form active so that an end user can use it, place the following syntax in your procedure:

```
Winform Show[Active] formname ;
```

Control is transferred to the form.

You can generate this syntax by dragging the form from the Project Explorer into the Procedure Editor or the Event Handler editor. (The form must be part of the procedure you are editing.)
Syntax: How to Display a Non-Persistent Form
To display a non-persistent form, place the following syntax in your procedure:

```
Wiform Show_And_Exit formname [;]
```

The form is displayed, but does not maintain a persistent connection with the server.

Syntax: How to Display a Form Without Making it Active
To display a form without making it active, place the following syntax in your procedure:

```
Wiform Show_Inactive formname [;]
```

Control passes to the next command in the procedure, but not to the form.

The Active Form
When a form is active, end users can fire events in the form, such as clicking a button or typing into a field. These events invoke event handlers, such as Maintain functions, JavaScript™ functions, VBScript functions, or web links.

When an event that is handled by a function is fired, the function is performed. Control returns to the server until the function is done, then the application displays a new browser page.

The Non-Persistent Form
The code that differentiates between a persistent and non-persistent form is the code you use to display your form. If you use the default code for Maintain, Winform Show, you create a persistent form. If you use the code that is embedded in the comment text at the top of each new form, Winform Show_And_Exit, you create a non-persistent form.

Forms in Maintain applications that were built in WebFOCUS Maintain Release 4.2 Version 1 or earlier are, by default, persistent. When a user accesses a persistent form, an agent from your server is accessed and sends the appropriate information to the browser. After the user has the form, the WebFOCUS Server agent waits for further instructions while the user makes choices on the form. That agent is dedicated to that user until the user closes the application. This method is appropriate if your application requires navigation from form to form.

With a non-persistent form, when the user accesses a non-persistent form, your server sends the appropriate information to the user, then disconnects the connection. The WebFOCUS Server agent is no longer dedicated to that user, and is available for another user to access. This method is appropriate for single form applications or for the last form in an application that uses multiple forms.
Closing Forms at Run Time

Maintain has several commands for closing forms at run time:

- The Winform Close form and self.WinClose commands close a form and return control to the command following the command that opened the form.

  **Note:** You can only issue self.WinClose in the Event Handler editor for a form or one of its controls.

- The Winform Close_All command closes all forms.

When you close a form, control returns to the command directly after the command that initially displayed the form, as shown in the diagram below. No additional commands should be placed immediately following the Winform Close command.
Suppose you have four forms: Form1, Form2, Form3 and Form4. Form1 opens Form2, Form2 opens Form3, and Form3 opens Form4. If you close Form2 while you are in Form4, WebFOCUS Maintain closes Form2, Form3, and Form4, and returns to Form1, as shown in the following diagram.

Winform Close closes every form in its chain.

**Syntax:** **How to Close a Form**

To close a specific form, place the following syntax in your procedure:

```
Winform Close [form] [;;]
```

or

```
self.WinClose();
```

These commands close the specified form and the chain of forms that may have been opened from this one. It returns control to the point just following the Winform Show command that displayed the form.
**Note:**

- If you do not specify a form, Winform Close closes the currently active form.
- The `self.WinClose` syntax only works in the Event Handler editor for a form or one of its controls.
- If you use one of these commands to close the last form in your application, thus closing the application, you receive an *Invalid Response* message from your browser. This behavior is expected. You should use the `self.WinExit();` syntax to exit your application.

**Tip:** You can generate the `self.WinClose` syntax in the Event Handler editor by clicking the `Close Form` button.

**Caution:** Closing a form returns control of the application to the command directly after the Winform Show form command that initially displayed the form. No additional commands should be placed immediately following the Winform Close command.

### Syntax: How to Close All Forms

To close all forms, place the following syntax in your procedure:

```plaintext
Winform Close_All [;]
```

### Manipulating Form Properties

The Maintain language contains three commands to manipulate the properties of your form:

- The Winform Get command enables you to assign the value of a property to a variable.
- The Winform Set command enables you to set the value of a form property (except for colors).

**Note:** If the form is not active, you must issue `Winform Show_Inactive form` before setting the form property.

- The Winform Reset form command enables you to reset all the properties of a form and its controls to the original settings.
Syntax: **How to Determine the Value of a Form Property**

To determine the value of a form property and assign it to a variable, use the following syntax

\[ \text{Winform Get } \textit{form.property INTO variable ];} \]

where:

- \textit{form} is the name of the form.
- \textit{property} is the name of the property.

Syntax: **How to Set the Value of a Form Property**

To set the value of a form property (except for colors), use the following syntax

\[ [\text{Winform Show_Inactive form;}] \text{Winform Set } \textit{form.property TO setting ];} \]

where:

- \texttt{Winform Show_Inactive} is required if the form is not the active form.
- \textit{form} is the name of the form.
- \textit{property} is the name of the property.

You cannot set the values of all properties. To see which properties you can set, see *Form and Control Properties Reference* on page 267.

**Tip:** You can generate this syntax by selecting the property in the property sheet and dragging it to the Procedure Editor or the Event Handler editor. Note that you cannot drag properties that you cannot set at run time.

For information on how to set color properties, see *Defining Colors for Your Form and Controls* on page 260.
**Syntax:  How to Reset a Form**

To reset a form and its controls to their original properties, place the following syntax in your procedure:

```plaintext
Winform Reset formname [;]
```

All selectable controls, such as list boxes, check boxes, and radio buttons, return to their default selections.

**Exiting Your Application at Run Time**

The Maintain language has two commands for exiting an application:

- The Winform Show_And_Exit form command exits the application and displays a final form.
- The `self.WinExit();` command, available only in the Event Handler editor, exits the application.

For more information on the Winform Show_And_Exit form command, see *How to Display a Non-Persistent Form* on page 116.

**Syntax:  How to Close the Application**

To close the application, place the following syntax in the Event Handler editor:

```plaintext
self.WinExit();
```

**Note:**

- The `self.WinExit` syntax only works in the Event Handler editor for a form or one of its controls.
- WebFOCUS Maintain ignores any commands after this one.

**Tip:** You can generate the `self.WinExit` syntax in the Event Handler editor by clicking the **Close Form** button 🗑️.
How WebFOCUS Maintain Saves Your Forms

When creating a Maintain project using the MDE, most of the logic that controls the appearance and behavior of the forms is not visible. In the procedure file (.MNT) related to the form, WINFORM SHOW is the only thing that is visible. A separate resource file (.WFM or WINFORM) that supplies the run-time form information is created during the deploy process from the internal form code stored in the original .MNT file at design time. This internal form code is not actually used at run time, and once the .WFM file is created the code can be kept in remotely deployed .MNT files by selecting an option in the Deploy wizard. For more information on this option, see Partitioning and Deploying Project Files in the Creating Reporting Applications With Developer Studio manual.

**Note:** The development copy of a Maintain project with forms will always contain the source code of the forms in the .MNT files. This code should never be manually edited. At compile time, both the .MNT code and the .WFM resource code are built into a run-time procedure (the .FCM, or FOCCOMP file).

Using a Driver Procedure

One of the most common errors in a WebFOCUS Maintain application is opening forms without ever closing them. As each form is opened in a WebFOCUS Maintain application, it is stacked on top of the previous form. Since each form is displayed in a separate browser page, it is easy to open more and more forms without ever closing them.

This leads to a Stack Overflow error. This error indicates that you have opened too many forms and never closed any of them. Your host platform cannot possibly handle holding that many in memory.

To avoid this problem, we recommend having only one form open at a time. When you open one form, close the previous one. Placing the following syntax in your procedure does not work because WebFOCUS Maintain ignores any code after a Winform Close statement. After closing FORM1, control returns to the command after the command that opened FORM1.

```
Winform Close FORM1
Winform Show FORM2
```

To solve this problem, you can implement a driver procedure.

The theory behind the driver procedure is simple. Have a variable (for example, NEXT_FORM) which contains the name of the next form to open. The driver procedure is just a REPEAT loop which opens the form indicated by NEXT_FORM. When you are done with NEXT_FORM, you set NEXT_FORM to contain the name of the next form that you want to open before coding the Winform Close form. This continues until you EXIT the application, or until NEXT_FORM equals EXIT or some other ending string.
**Example:** **Coding a Driver Procedure**

Here is an example of some of a driver procedure:

```plaintext
MAINTAIN

CASE TOPCASE
  COMPUTE NEXT_FORM/A66 = 'MAINMENU';
  PERFORM DRIVER
ENDCASE

CASE DRIVER
  REPEAT WHILE NEXT_FORM NE ' ';
    IF NEXT_FORM EQ 'MAINMENU'
      THEN Winform Show MAIN_MENU_FORM
    ELSE
      IF NEXT_FORM EQ 'UPDATE'
        -* call another procedure to show a form
          THEN CALL UPD_PROC FROM NEXT_FORM INTO NEXT_FORM
        ELSE
          IF NEXT_FORM EQ 'DELETE'
            THEN Winform Show DELETE_FORM
      ENDIF
  ENDREPEAT
ENDCASE
```

Notice that TOPCASE sets NEXT_FORM to be MAINMENU (this is the first form that we show). This form has two buttons: Update and Delete. These two buttons, when clicked, execute the functions UPDATE_CASE and DELETE_CASE, respectively:

```plaintext
CASE UPDATE_CASE
  COMPUTE NEXT_FORM = 'UPDATE'
  Winform Close MAIN_MENU_FORM
ENDCASE

CASE DELETE_CASE
  COMPUTE NEXT_FORM = 'DELETE'
  Winform Close MAIN_MENU_FORM
ENDCASE
```

Notice in the above cases, we set NEXT_FORM and then close MAIN_MENU_FORM. The driver loop takes care of showing the next form. Only one form is ever open.
The driver loop handles UPDATE and DELETE in two different ways. DELETE just shows another form (DELETE_FORM), but UPDATE actually calls another procedure called UPD_PROC. Look at UPD_PROC:

```
MAINTAIN FROM NEXT_FORM INTO NEXT_FORM

CASE TOPCASE
   COMPUTE
      NEXT_FORM/A10='MAINMENU';
      LOCAL_NEXT_FORM/A10='UPD_FORM1';
   PERFORM MINI_DRIVER
ENDCASE

CASE MINI_DRIVER
   REPEAT WHILE LOCAL_NEXT_FORM NE ' ';
      IF LOCAL_NEXT_FORM EQ 'GOAWAY'
         THEN GOTO EXITREPEAT
      ELSE
         IF LOCAL_NEXT_FORM EQ 'UPD_FORM1'
            THEN PERFORM Winform_UPD_FORM1
         ELSE
            IF LOCAL_NEXT_FORM EQ 'UPD_FORM2'
               THEN PERFORM Winform_UPD_FORM2
         ENDREPEAT
     ENDREPEAT
ENDCASE
```

The initial procedure passed NEXT_FORM to this procedure from the CALL statement. This way, when UPD_PROC finishes executing and returns to the initial procedure, it knows where to go. Also, UPD_PROC defines a LOCAL_NEXT_FORM variable and a mini driver function within UPD_PROC, so we can control the form flow locally to this procedure too.

Clicking the Delete button on UPD_FORM2 triggers the following function and goes directly to DELETE_FORM in the driver.

```
CASE DELETE_FORM
   COMPUTE NEXT_FORM = 'DELETE';
      LOCAL_NEXT_FORM = 'GOAWAY';
   Winform Close UPD_FORM2
ENDCASE
```

You can use this driver technique to control the flow of your entire application. It is incredibly useful and guarantees that you do not have a Stack Overflow problem in your application, because you only have one form open at any given time.

It is very easy to write one of these driver loops if you diagram your Form flow.
Defining Events and Event Handlers

After you define the appearance of your form, you must define the event handlers. Event handlers are the actions that occur when end users perform certain events (such as entering text in a field or clicking a button).

This topic describes how to define events and how to bind an event to a control.

In this chapter:

- Using the Event Handler Editor
- Events
- Using a Maintain Function as an Event Handler
- Using Script Functions as Event Handlers
- Using Web Links as Event Handlers

Using the Event Handler Editor

You define event handlers for specific events using the Event Handler editor.

Here is a brief overview of how to define events:

1. Open the Event Handler editor. For more information, see How to Open the Event Handler Editor on page 126.

2. Select a control or a form.

3. Select an event that happens to that form or control (for example, Click).

4. Select the action that occurs. The editor offers six choices:
   - Using a Maintain Function as an Event Handler on page 131.
   - Using Script Functions as Event Handlers on page 132.
   - Using Web Links as Event Handlers on page 143.
   - Close the form (the Maintain function button must be selected). For more information, see Closing Forms at Run Time on page 117.
Procedure: How to Open the Event Handler Editor

To open the Event Handler editor, do one of the following:

- In the Project Explorer, right-click the form and in the shortcut menu, click *Edit event handlers*. The form will be selected in the Event Handler editor.

- In the Form menu in the Form Editor, click *Edit event handlers*.

- In the Form Editor, click the *Edit event handlers* button on the Layout toolbar.

- In the Form Editor, right-click a control. In the shortcut menu, click *Edit event handlers*. (The control will be selected in the list of controls in the Event Handler editor.)

  **Tip:** If the control is a button, you can double-click the button to open the Event Handler editor.

- In the property sheet, for a particular control, click the *Events* tab. Then, double-click any event in the list of events. When the Event Handler editor opens, that event will be selected.
**Reference: Event Handler Editor**

The Event Handler editor is where you define the event handlers that are invoked when events occur in your application. An example of the Event Handler editor is shown in the following image.

This editor includes the following options:

**List of controls**
Contains the controls on the form, as well as the form itself. One of these controls will already be selected when you open the Event Handler editor, but you can select another one if you wish.

**List of events**
Contains the possible events that can happen to the control or form you selected. For a description of these events, see *Events* on page 129.

If you click the *Maintain function* button, your event handler will be a Maintain function.

After clicking the *JavaScript* button, your event handler becomes a JavaScript function.
If you click the VBScript button, your event handler will be a VBScript function.

The Web link button allows to turn your event handler into a web link.

The Delete button deletes code for the existing event handler.

If you click the Close form button, the code for closing your form (self.WinClose();) is inserted.

**Note:** This code should be the last line of code in the event handler. Do not enter any code after this syntax. It will not be executed.

If you use this command to close the last form in your application, thus closing the application, you will receive an Invalid Response message from your browser. This behavior is expected. You should use the Self.WinExit(); syntax to exit your application.

If you click the Close project button, the code for closing your application (self.WinExit();) is inserted.

**Note:** This code should be the last line of code in the event handler. Do not enter any code after this syntax. It will not be executed.

**Event handler code area**

If you selected the Maintain function, JavaScript, or VBScript button, this area is where you enter the code that defines your event handler.
Events

WebFOCUS Maintain includes the following events:

Blur

Occurs when an end user leaves a control that was in focus. The following controls can use this event: button, check box, combo box, edit box, image, list box, multi-edit box, radio button.

**Note:** If a control has focus and an end user clicks anywhere else (even on a button) the Blur event on the first control is executed first, before the Click event on the next control.

Check

Occurs when an end user selects a check box.

Change

Occurs when an end user changes a value. The following controls can use this event: combo box, edit box, image, list box, multi-edit box, radio button.

Click

Occurs when an end user clicks a form or control. The following controls can use this event: button, edit box, form, group box, HTML Object, HTML Table, image, line, multi-edit box, text list box, and combo box.

ClickArea

Occurs when an end user clicks an image with an image map, where Area is the name of the area in the image map. You will see a ClickArea event for each area in your image map.

For example, if your image control uses a map with two areas, named Fish and Boat, you will see the events ClickFish and ClickBoat for your image.

ClickLink

Occurs when an end user clicks an HTML Table. This is a special event that enables you to use syntax to determine which row, column, or cell the end user clicked.

ClickMenuItem

Occurs when an end user clicks on an item in a menu, where MenuItem is the name of the menu item. You will see a ClickMenuItem event for each item in your menu.

For example, if your menu has the items Open and Exit, you will see the events ClickOpen and ClickExit for your image.
Events

**Close**

Occurs when a form is closed.

**DoubleClick**

Occurs when an end user double-clicks a form or control. The following controls can use this event: button, edit box, form, group box, HTML Object, HTML Table, image, line, list box, multi-edit box, and text.

**Focus**

Occurs when control gets focus, for example, when an end user selects or tabs to a control. The following controls can use this event: button, check box, combo box, edit box, image, list box, multi-edit box, and radio button.

**KeyPress**

Occurs when an end user presses a key while an edit box, form, or multi-edit box is in focus.

**MouseDown**

Occurs when an end user places the mouse pointer on a control or form and presses the mouse button. All controls can use this event.

**MouseMove**

Occurs when an end user moves the mouse while in a control. All controls can use this event.

**MouseOut**

Occurs when an end user moves the mouse pointer off a control. All controls can use this event.

**MouseOver**

Occurs when an end user moves the mouse pointer onto a control. All controls can use this event.

**MouseUp**

Occurs when an end user releases the mouse button. All controls can use this event.


**Note:** The event handler for this event can only be a JavaScript or VBScript.

These events occur only in the Grid control. For more information, see Events Available in a Read/Write Grid on page 199.

Open

Occurs when a form is opened or shown.

Note: You can only use a Maintain function as an event handler for this event. You cannot use JavaScript functions, VBScript functions, or web links (due to unpredictable behavior). To execute a JavaScript or VBScript function or display a web link when a form opens, use an HTML Object.

UnCheck

Occurs when an end user deselects a check box.

Common Combinations of Events and Controls

The following are some common combinations of events and controls. Your application will probably include many of these, but it is not limited to them.

- Clicking a button or image.
- Opening a form.
- Closing a form.
- Blurring (or leaving) an edit box. You can execute a validation procedure to check that the edit box contains valid data.

Using a Maintain Function as an Event Handler

The most common event handler is a Maintain function.

We recommend that, whenever possible, you create a Maintain function using the Project Explorer, and then invoke this Maintain function from the Event Handler editor. Doing so has the following advantages:

- You can reuse the Maintain function.
- It is easier to debug your application, since you do not have to open the Event Handler editor to edit your code. Instead, you can edit functions from the Project Explorer.
**Procedure: How to Execute a Maintain Function From the Event Handler Editor**

1. Open the Event Handler editor.
2. Select a control and an event from the drop-down lists.
3. Click the *Maintain function* button.
4. Type the following code in the box:

   ```
   PERFORM function();
   
   where:
   
   function
   
   Is the name of the function (or case) you want to call.
   ```

**Tip:** Instead of typing the above statement, you can drag the function from the Project Explorer into the Event Handler editor.

When an end user causes the event to happen to the control (for example, a button being clicked), the application executes the function.

**Using Script Functions as Event Handlers**

One of the components in a WebFOCUS Maintain project is a script. A script contains stand-alone functions, written in either JavaScript or VBScript for execution on the client. The script code can either be embedded in your form or you can use a pointer within the form to a script file located on a web server.

Why use scripts? JavaScript and VBScript functions can perform local validation, which means, they can validate user interaction on a form without having to return to the server.

For example, JavaScript and VBScript functions can do the following, all without connecting to the server:

- If your application requires users to enter information and some of the fields are required, you can write a function that confirms that users have entered information in the required fields.
- If your application requires users to enter a credit card number, you can use a function to make sure the number is the correct length and is made up of digits 0 through 9.
Suppose your application requires that users change their passwords periodically. Most password changing utilities require users to enter their new password twice. You can use a function to verify that users entered the same password.

You can ensure that users do not lose data by exiting an application without saving. Of course, this validation can be done with the Maintain language, but using JavaScript or VBScript is an optimization that saves you a trip back to the server.

**About JavaScript and VBScript**

JavaScript is a small, object-based scripting language that you can use to create functions that can be embedded in HTML pages. When you execute your application, your browser will interpret the code and perform the actions required.

VBScript is similar to JavaScript, except that it is only supported in Microsoft Internet Explorer. All of our examples will use JavaScript. However, you can use VBScript to perform similar functions.

This help system document will show how to use several JavaScript functions. However, it is not a complete guide to JavaScript.

**Tip:** JavaScript and VBScript are case-sensitive. Make sure you are using the correct capitalization, especially when referring to controls in a form or to Maintain functions.

**Using Script Functions in Your Project**

There are two ways to write JavaScript and VBScript functions:

- Enter the code for your script function directly in the Event Handler Editor. For more information, see *How to Assign a JavaScript or VBScript Function in a Script Library to an Event* on page 134.
Use the Object Explorer to create your script file and enter the code for your functions there. If you are coding an event for a control, open the Event Handler editor and enter the name of the function you want to execute. This method enables you to create functions you can use throughout a project. If you are coding a JSGridOnLoad function, embed the script directly onto the form by dragging it from the Object Explorer onto the form in the MDE. For more information, see How to Assign a JavaScript or VBScript Function in a Script Library to an Event on page 134 and Managing Script Libraries on page 142.

For JavaScript functions only: You can include customized JavaScript functions that will apply to all of your Maintain applications (not just a specific project), by including them in the mntonload.js file, located in the ibi_html\javaassist\ibi\html\maint directory of your run-time configuration. For more information, see Customizing Key Functions and JavaScript for WebFOCUS Maintain Applications on page 348.

Procedure: How to Write a JavaScript or VBScript Function in the Event Handler Editor

1. Open the Event Handler editor.
2. Select a control and an event from the drop-down lists.
3. Click the JavaScript button or VBScript button.

Type the code for your function in the box.

When the event occurs (for example, a button being clicked), the browser will execute the function.

Procedure: How to Assign a JavaScript or VBScript Function in a Script Library to an Event

If you code a function in a script library, you must then assign this function to an event. For more information on script libraries, see Managing Script Libraries on page 142.

1. Create the script library and code the function in it.
2. Open the Event Handler editor.
3. Select a control and an event from the drop-down lists.
4. Click the JavaScript button or VBScript button.
5. Enter the name of the JavaScript or VBScript function that you would like to call in the box using the following syntax:

   function(parameters);
6. Drag the script library that contains the function from the Explorer to the open form in which you are using it. (This is the Object Explorer, not the MDE Project Explorer.) This ensures that WebFOCUS Maintain includes the script only in the page that is using it. Select either to Embed script or Link script. Note that linked scripts must be assigned a web server location during the deployment phase.

**Tip:** If you do not see your script files listed for your project in the Explorer, you may need to add a filter for the file type. You can do this from the Edit Filters tab in the properties sheet for the project.

When the event occurs (for example, a button being clicked), the browser will execute the function.

### Using Script Functions for Validation

When you create a function to perform local validation, the function generally performs three tasks:

- The function must communicate with the end user, either to inform the end user of some important information, to ask for information, or to confirm some action.

  Three JavaScript functions that you can use to perform these tasks are alert, prompt, and confirm. For more information, see *How to Use the JavaScript Alert Function* on page 136, *How to Use the JavaScript Confirm Function* on page 136, and *How to Use the JavaScript Prompt Function* on page 137.

  The dialog boxes that these three functions generate are browser-specific. WebFOCUS Maintain has no control over them and there is no way to customize them from JavaScript (or VBScript). For example, you cannot change the heading of the Prompt box or the OK button on a Confirm box.

- The function must test a value, either one that it has prompted the end user for, or one that is already indicated on the form. A function can access any control on a form (buttons, images, edit boxes) using the syntax `form.control.property`, except for text controls (they are just static text and have no dynamic value).

- If the validation test fails, the function leaves end users in the form from which they executed it. If the validation test succeeds, the function must return control to the WebFOCUS Maintain application, usually to a Maintain function. It does this by using the IWCTrigger function, provided by Information Builders for use with any WebFOCUS Maintain application. For more information, see *How to Use the IWCTrigger Function to Call a Maintain Function From Your Script Handler* on page 137.
**Tip:** JavaScript and VBScript are case-sensitive. Make sure you are using the correct capitalization, especially when referring to controls in a form or to Maintain functions.

**Syntax:**

**How to Use the JavaScript Alert Function**

The JavaScript alert function displays a pop-up window with an OK button. The syntax is

```javascript
alert("text");
```

where:

- **text**
  - Is informational text that is displayed in the pop-up window. You can enter a variable name here (omit the quotation marks).

**Note:** We are providing the syntax for this function here, but see your JavaScript documentation for a complete description of this language.

**Syntax:**

**How to Use the JavaScript Confirm Function**

The JavaScript confirm function displays a pop-up window with a question, and OK and Cancel buttons. If the end user clicks OK, the function returns TRUE. If the end user clicks Cancel, it returns FALSE. The syntax is

```javascript
value = confirm("text");
```

where:

- **value**
  - Is the name of the variable set to either TRUE or FALSE, depending on what the end user clicks.

- **text**
  - Is the text of the question you are asking the user. It should be a yes or no question. You can enter a variable name here (omit the quotation marks).

**Note:** We are providing the syntax for this function here, but see your JavaScript documentation for a complete description of this language.
**Syntax:** How to Use the JavaScript Prompt Function

The JavaScript prompt function displays a pop-up window with a message, an edit box, and an OK button. The function returns the value the end user enters in the text box. The syntax is

```javascript
value = prompt("text", defaultvalue);
```

where:

- **value**
  - Is the name of the variable to contain the value the end user enters in the edit box.

- **text**
  - Is the text that appears in the pop-up window. You can enter a variable name here (omit the quotation marks).

- **defaultvalue**
  - Is an optional parameter that places a default value in the text box.

**Note:** We are providing the syntax for this function here, but see your JavaScript documentation for a complete description of this language.

---

**Syntax:** How to Use the IWCTrigger Function to Call a Maintain Function From Your Script Handler

The syntax for the IWCTrigger function is

```javascript
IWCTrigger("functionname", "parameter")
```

where:

- **functionname**
  - Is the name of the Maintain function you are calling.

**Note:** Scripts are case-sensitive, so you must specify the name using the same uppercase and lowercase letters that you used to name the function in the Maintain procedure.

- **parameter**
  - Is an optional parameter that you can pass to the Maintain function.

The called function can retrieve this parameter using the following syntax

```javascript
formname.Triggervalue
```
where:

formname

Is the name of the form.

Note:

- IWCTrigger can be used in the same way from VBScript. IWCTrigger is a WebFOCUS Maintain-supplied script function for use in any WebFOCUS Maintain application.
- If you use IWCTrigger in a script library, make sure that the Maintain function you are calling is in the same procedure in which you are using the script library.
- You can drag a Maintain function from the Project Explorer into a script being edited in the Script Editor, and WebFOCUS Maintain will generate the IWCTrigger syntax for you.

Example: Validating End User Input With JavaScript

Suppose you have an application that prompts an end user to enter a numerical value (in the OrderNo edit field) and click a Submit button. The application then executes a Maintain function called DoProcess. The Click event in the Submit Button is assigned to the Maintain function DoProcess.

The name of the form is OrderStatus and the name of the edit box into which the end user enters a number is OrderNo, as shown in the following image.
However, before executing DoProcess, you want the Submit button to execute a JavaScript function (called test_it) that will test the value in OrderNo. The value in OrderNo must not be blank, and can only contain the digits from 0 to 9.

Create a new JavaScript library by right-clicking the project folder in the Explorer, clicking New in the shortcut menu, and then clicking Project file. In the Create a New File dialog box, give your JavaScript library a name.

Then open the JavaScript library and enter the code for the test_it function in your script library.

Here is the JavaScript code for test_it:

```javascript
function test_it()
{
    okay=1
1.  fldvalue=document.OrderStatus.OrderNo.value
2.  if (fldvalue == '')
      alert("Please enter some data!");
    else
      {
3.    for(var i=0;i<fldvalue.length;i++)
        {
            var ch=fldvalue.substring(i,i+1)
4.      if (ch<"0" || "9"<ch)
        {
5.        if (ch != ".")
          {
              okay=0
              alert("Please enter only numeric data!");
              break
          }
        }
5.    }
6.  if (okay==1)
      IWCTrigger("DoProcess")
    }
}
```

1. This line of code sets `fldvalue` to be the value of OrderNo on OrderStatus on this HTML document.
2. The first `if` test checks to see if `fldvalue` is blank. If so, it pops up a dialog box saying, “Please enter some data!”
3. If `fldvalue` is not blank, then test_it enters a `for` loop that lasts for the length of `fldvalue`.
4. Within the `for` loop, test_it tests each character in `fldvalue`. 
5. If test_it finds a character that is less than 0 or greater than 9 and that character is not a decimal, then it sets okay to 0, alerts the user to “Please enter only numeric data!” and breaks out of the for loop.

6. The last if statement just checks to see if test_it has gotten through all of the for loop with okay still equal to 1, in which case it can execute DoProcess (using IWCTrigger).

With the function complete, you must now to assign it to the Submit button in place of DoProcess. See Executing a JavaScript Function From a Button on page 140.

**Example:** Executing a JavaScript Function From a Button

In Validating End User Input With JavaScript on page 138, you started with an application that prompts an end user to enter a numerical value (in the OrderNo edit field) and click a Submit button. The application then executes a Maintain function called DoProcess, which is the function assigned to the Click event.

You created a JavaScript function named test_it that tested the value in OrderNo to make sure it was numeric. With the script complete, you must now assign it to the Submit button in place of DoProcess, as shown in the following image.

1. Open the OrderStatus form.
2. Drag Script1 from the Explorer to the open OrderStatus form. This ensures that WebFOCUS Maintain includes this script only in OrderStatus.
3. Open the Event Handler editor for the form OrderStatus.
4. From the list of controls on the form, select SubmitButton.
5. From the list of events that could happen to that control, select Click.

6. Click the *JavaScript* button to indicate that clicking *SubmitButton* would execute a JavaScript.

7. Type `test_it();` in the box, as shown in the following image.

Now, when end users executing this application click the *Submit* button, their browsers execute the *test_it* function.

**Using Script Functions With HTML Tables**

There is a special case when an end user executes a JavaScript function by clicking an HTML Table. In this case, you can use code in your function to identify the row number, column number, or value of the cell clicked by the end user.

**Syntax:** **How to Determine Row Number, Column Number, or Cell Value When an HTML Table Is Clicked Using a Script**

If your function is executed by an end user clicking on an HTML Table, then you can use special syntax in your function to determine what part of the HTML table the end user clicked.

If you want to determine the row number, use:

```javascript
document.formname.tablename_ClickRow.value
```

The header row returns 0, and the first data row returns 1.

If you want to determine the column number, use:

```javascript
document.formname.tablename_ClickColumn.value
```

If you want to determine the contents (text) of the cell, use:

```javascript
document.formname.tablename_Value.value
```
Using Script Functions to Turn Secure Sockets Layer On and Off

You also use JavaScript functions to turn a Secure Sockets Layer on and off (which you might use when you want to perform private communications, such as transmitting credit card information for E-commerce applications).

Using Script Functions to Make Controls Visible

You can use script functions to dynamically make a control visible.

**Syntax:** How to Make a Control Visible

```javascript
document.Formname.Controlname.style.visibility="visible";
```

where:

- **Formname**
  - Is the name of your form.

- **Controlname**
  - Is the name of your control.

Managing Script Libraries

You manage script libraries in the Explorer. However, you must set up the Explorer to view script files first. For more information, see *How to View New File Types in the Explorer* on page 35.

**Note:** The default extension for JavaScript files is .js and the default extension for VBScript files is .vbs.

From the Explorer, you can:

- Create script libraries. (For more information, see *How to Create a Project Component* on page 29).
- Rename script libraries. (For more information, see *How to Rename a Project or Project Component* on page 35).
- Remove script libraries from the project. (For more information, see *How to Remove a Project Component From a Project* on page 36).
- Delete script libraries. (For more information, see *How to Delete a Project or Project Component* on page 36).
- Open script libraries for editing in the Script Editor.
**Procedure:** How to Edit a Script Library

1. Right-click the script library.
2. In the shortcut menu, click *Edit source*.

WebFOCUS Maintain opens the Script Editor where you can make changes to your script library. For more information, see *Using Script Functions in Your Project* on page 133.

Your next step is to associate your script library with a form. See *How to Associate a Script Library With a Form* on page 143.

**Procedure:** How to Associate a Script Library With a Form

To ensure that a form will contain the script functions it uses, you can associate the script library with the form.

1. In the Explorer, select the script library.
2. Drag the script library to the open form in the Form Editor. Make sure you drag the script library to an empty space on the form, not on top of a control.

Your last step is to assign the functions you have coded in your script library to events. For more information, see *How to Assign a JavaScript or VBScript Function in a Script Library to an Event* on page 134.

**Running and Debugging JavaScript and VBScript Functions**

WebFOCUS Maintain can parse the syntax of your script functions to see if they are correct. If WebFOCUS Maintain discovers incorrect syntax, you will see an exclamation mark in the left margin next to the incorrect line of syntax and an error message in the General tab of the Output window. Double-click the line in the Output window to open the script to the offending line.

Keep in mind that JavaScript and VBScript are case-sensitive, so one common cause of errors is referring to functions or controls by the correct name, but in the wrong case.

**Using Web Links as Event Handlers**

One of the components of your project is a Web link. A web link is an object that contains a valid URL, such as a webpage, email address, or FTP server (basically anything that can be typed into a Location box in the browser). A web link can also be an HTML file.

You create and use a web link as an event handler in your application. For example, in the event handler editor, you might create a web link that maps to a webpage, assigning it to an event that occurs when a user clicks a button. During run time, clicking this button opens the webpage you defined in your web link.
You can also use a web link HTML file as a Help file using the Help property. For more information, see *Assigning Help to Your Forms and Controls* on page 263.

You can also launch a web link into a frame using the frame control. For more information, see *Using Frames* on page 191.

Here are some ways you might use web links:

- Create a button on every form that jumps to the homepage of your company.
- Create a Suggestions button. Clicking this button would enable end users to compose and send email to the staff responsible for administrating the application. Email links require that the browser of the end user supports Internet mail services.
- Create a button that downloads instructions for using your application.
- Create a button that runs a WebFOCUS report based on values supplied at run time.

When you define a link, you give it a name. Because you refer to a link by its name and not its URL, it is easy to maintain. For example, if you refer to the same web link from several places in your project and you need to update the URL, you do so only once in the Link Editor, and the change will be reflected wherever you use that web link in the project.

**Procedure: How to Create a Web Link**

1. In the Event Handler editor, click the *Web link* button.
2. In the Web Link dialog box, click *New*.
3. In the Link Editor dialog box, type a name for the web link in the Name box.
4. Depending on the type of web link you are creating, you can type a URL for the web link in the Universal Resource Locator (URL) box or click the ellipsis button to open the URL Wizard.

   or

   Select an HTML file in the list of HTML file resources in your project. If the one you want to select is not there, click New to open the Resource Wizard.

   For more information on defining web links, see *How to Use the URL Wizard to Define an HTTP Web Link* on page 147, *How to Use the URL Wizard to Define an FTP Web Link* on page 147, *How to Use the URL Wizard to Define a WebFOCUS Report as a Web Link* on page 148, and *How to Use the URL Wizard to Define an Email Web Link* on page 148.

5. Optionally, type a description for the web link by clicking the *Description* tab and entering a description.
6. Click OK when you are done.

**Example: Creating and Using a Link**

Suppose you decide to include a button in a user application that links to the Information Builders home page: www.informationbuilders.com.

1. Open the Form Editor and create a button with the name IBIWeb and the title Information Builders on the web, as shown in the following image.

2. Double-click the IBIWeb button to open the Event Handler editor.

3. In the Event Handler editor, select the Click event, and then select the Web Link button. WebFOCUS Maintain opens the Web Link dialog box, as shown in the following image.

4. Create a web link by clicking New.
5. In the Link Editor dialog box, name your web link *IBIhome* and enter the URL http://www.informationbuilders.com, as shown in the following image.

6. Enter a description of the web link in the Description tab, as shown in the following image.
7. Click OK in the Link Editor and the Web Link dialog box. The Event Handler editor will look similar to the following image.

8. Close the Event Handler editor and save your project.

9. When you deploy and run the application, clicking Information Builders on the Web will open the home page of Information Builders.

**Procedure: How to Use the URL Wizard to Define an HTTP Web Link**

1. In the Link Editor, click the ellipsis button.
2. In the URL Wizard, select http or https: - Hypertext Transfer Protocol and click Next.
3. Enter the web server name.
4. If you are connecting to a secure server, select Make this a secure hyperlink (https:).
5. Click Next.
6. Optionally, if you want to link to file, directory, or alias on the server, enter its name in the Object path name box.
7. Click Finish.

**Procedure: How to Use the URL Wizard to Define an FTP Web Link**

1. In the Link Editor, click the ellipsis button.
2. In the URL Wizard, select ftp: - File Transfer Protocol and click Next.
3. Enter the domain name of your FTP Server and click Next.
4. Optionally, if you want to link to a file, directory, or alias on the FTP server, enter its name in the Object path name box.

5. Click Finish.

**Procedure:** How to Use the URL Wizard to Define a WebFOCUS Report as a Web Link

1. In the Link Editor, click the ellipsis button.

2. Select the WebFOCUS report radio button in the URL Wizard dialog box and click Next.

3. Select from the drop down list or type in the domain name of your WebFOCUS Server. Click Next.

4. Select the name of a WebFOCUS report from the list and click Next.

5. Bind a value to each parameter. To bind an existing parameter, double-click on that parameter in the Parameter column. To add a new parameter, click the Add to list button. The Enter a Parameter dialog box will display.

6. In the Enter a Parameter dialog box, specify a value for your parameter. To bind a parameter manually, select the As Entered Below radio button. Type the value you wish to specify in the Value text box.

7. To bind a parameter from a control, select the From a control radio button. Specify the Form name and Control name in the drop-down text boxes provided.

8. Click OK.

9. When you have specified values for all parameters in the report, review the display in the URL Wizard and click Finish. WebFOCUS Maintain generates the complete URL and puts it in the URL box.

**Note:** The URL Wizard does not parse the initial amper variable in a -SET statement. Only if the -SET statement contains amper variables to the right of the equal sign will that parameter appear in the URL Wizard dialog box.

**Procedure:** How to Use the URL Wizard to Define an Email Web Link

1. In the Link Editor, click the ellipsis button.

2. In the URL Wizard, select mailto: - MailTo Protocol and click Next.

3. Define the recipients who will receive your email message and click Next.

4. If you wish, define the CC recipients who will receive your email message and click Next.

5. Enter a subject and body for your email message and click Finish.
**Procedure: How to Add an HTML File to Your Project as a Resource**

1. In the Link editor, select the *HTML Resource* radio button and click *New*.

   WebFOCUS Maintain opens the Resource Wizard, a series of windows that guide you through adding an HTML file to your project.

2. Enter the path to the HTML file. If you prefer to browse for the HTML file, click the ellipsis button.

   The Resource Wizard will automatically display the HTML file. Click *Refresh* to update the display.

   Then click *Next*.

3. Specify a unique name for your HTML file.

   Then click *Finish*.

4. You may see a dialog box informing you that WebFOCUS Maintain needs to make a copy of the HTML file in the project directory. Click *OK*.

**Procedure: How to Use a Web Link as an Event Handler**

1. Open the Event Handler editor.

2. Select a control and an event from the lists.

3. Click the *Web link* button.

4. Select the web link from the list of web links.

5. Enter a target name to define the name of a frame, or select one of the predefined frames (_self, _parent, or _top).

6. Determine whether you want to open the web link in a new window or a frame in the existing window. If you select new window, you can specify options for this new window by clicking *Options*.

   When the event occurs (for example, a button being clicked), the browser will open the web link.

**Procedure: How to Edit a Web Link**

1. Open the Event Handler editor for the control and event assigned to the Web Link.

2. Click the *Web link* icon to open the Web Link dialog box.

3. Click the *Edit* button.

4. Make any changes in the Link Editor dialog box.
5. Click OK when you are done.

**Procedure: How to Delete a Web Link**

1. Select the web link in the Project Explorer.
2. Right-click the web link, and in the shortcut menu, click *Delete*.
   
   or

   Press the Delete key.

   or

   Click the *Delete* button on the General toolbar.

**Reference: Web Link Dialog Box**

When you indicate that you want to use a web link as an event handler in the Event Handler editor, WebFOCUS Maintain displays the Web Link dialog box, shown in the following image.

![Web Link Dialog Box](image)

This dialog box has the following fields:

**[list of available web links]**

Select the name of an existing web link in the list. *<None>* is the default.

**New**

Opens the Link Editor, where you can define a new web link.
Link Target Name

Specifies the name of a target in which you can launch your web link. You can type a name here, or select one of the following:

_self. Launches the web link in the current frame.

_parent. Launches the web link in the frame containing the current frame.

_top. Launches the web link in the highest frame.

Open link in

Select New window to open your web link in a new window. Select Frame to open your web link in a frame on the current form.

Options

If you open your web link in a new window, click this button to open the Window Options dialog box, where you can customize the appearance of this window.

Reference: New Window Options Dialog Box

The New Window Options dialog box, shown in the following image, determines the appearance of the new browser window in which your web link appears. You will only see this dialog box if you select a web link as an event handler, you select the Opens new window option, and you click the Options button.
This dialog box has the following fields:

**Directories**
- Determines if you can see the Links toolbar on the browser window.

**Location**
- Determines if the browser window displays a Location box where the end user can type new URLs.

**Menubar**
- Determines if the browser window displays a menu bar.

**Sizable**
- Determines if the end user can resize the browser window.

**Scrollbars**
- Determines if the browser window has scroll bars.

**Status**
- Determines if the browser window displays a status bar.

**Toolbar**
- Determines if the browser window displays a toolbar.

**Height**
- Determines the height of the browser window.

**Width**
- Determines the width of the browser window.

**Other options**
- Specifies other HTML options for the appearance of your new browser window. WebFOCUS Maintain will append this code to the end of its generated code.
**Reference:** Link Editor URL Tab: Specifying a Name and URL for a Web Link

When you create or edit a web link, WebFOCUS Maintain opens the Link Editor, as shown in the following image.

![Link Editor](image)

This dialog box includes the following options:

**Name**
Type the name by which you want to refer to this link.

**Universal Resource Locator (URL)**
Type the web address for the web link.

Opens the URL Wizard, which guides you through the process of defining a web link.

**HTML Resource**
Select the name of an HTML file from the list.

**New**
Opens the Resource Wizard, where you can define a new HTML file to your project.
Reference:  Link Editor Description Tab: Adding a Description to a Web Link

When you click the Description tab in the Link Editor, WebFOCUS Maintain opens a dialog box where you can enter a description for your web link, as shown in the following image.
Reference: **URL Wizard: Specifying What Kind of Web Link to Create**

When you open the URL Wizard from the Link Editor, you first see a window where you specify what kind of web link you want to create. WebFOCUS Maintain enables you to create four types of web links in the URL Wizard, as shown in the following image.
Reference: URL Wizard: Specifying Name of WebFOCUS Report

Use this window to specify the name of the WebFOCUS report that you want to execute from your application.

**WebFocus report name**

Enter the name of a WebFOCUS report here. This report must be on the execution path of the WebFOCUS Server to which you are connecting.
Reference:  URL Wizard: Specifying Parameters for a WebFOCUS Report

If you are defining a web link to a WebFOCUS report, you may need to specify values for parameters at run time. In WebFOCUS reports, these parameters are called amper variables. If you are defining a report that contains parameters, the URL Wizard will parse the report, and parameter names will appear in the Parameters column of this dialog box in the URL Wizard, as shown in the following image.

This dialog box includes the following options:

**Parameter**

Lists the parameters in the report. To edit the name or value for a parameter, double-click it here.

**Value**

Lists the values that you have supplied for the parameters.
Opens the Enter a Parameter dialog box, where you can create a new parameter and supply a value.

Deletes the selected parameter.

Moves the selected parameter up in the list.

Moves the selected parameter down in the list.

**Reference: URL Wizard: Enter a Parameter Dialog Box**

The Enter a Parameter dialog box provides the following options when the *From a Control* radio button is selected, as shown in the following image:

**From a control**

When selected, enables you to supply a parameter value from a control.
Parameter name

Displays the name of the parameter. If you are adding a parameter, you can name the new parameter here.

Please select a Form name

Enables you to specify the name of the form you wish to use to bind your parameter.

Please select a controller

Enables you to specify the control you wish to use as the value for your parameter.

The Enter a Parameter dialog box provides the following options when the From Value radio button is selected, as shown in the following image:

![Enter a Parameter dialog box]

From Value

When selected, enables you to manually specify a value for a parameter.

Parameter name

Displays the name of the parameter. If you are adding a parameter, you can name the new parameter here.

Value

Enables you to specify the value you wish to bind to the parameter.
Reference: URL Wizard: Specifying E-mail Addresses

If you are defining an email web link, use the URL Wizard, shown in the following image, to define the recipients and CC recipients of your email message.

This dialog box includes the following options:

To Addresses

Lists the email addresses to which this message will be sent. To edit an address, double-click it.

Opens the Email Address dialog box, where you can enter a new address.

Deletes the selected address.

Moves the selected address up in the list.
Moves the selected address down in the list.

**Reference:** Email Address Dialog Box

Use the Email Address dialog box, shown in the following image, to enter a new address.

![Email Address Dialog Box](image)

**Recipient's user name**

Enter the user name of your email recipient here.

**Domain name**

Enter the domain of your email recipient here, or select a domain you have already used in this project from the list.

WebFOCUS Maintain creates the email address by concatenating the user name of the recipient to the domain name, separated by the at (@) symbol.
Reference:  URL Wizard: Entering an Email Message Subject and Body

Use this window to enter an optional subject and body for your email message. The end user has the option of changing these fields at run time.
Developing and Using Controls

There are many different controls that you can use in your project forms. This topic provides information on using these controls, as well as determining and setting the values of control properties at run time, using the Winform command. There is also information on defining form and control attributes that are shared among many or all of the controls, as well as manipulating controls on your form.

In this chapter:

- Which Control Should You Use?
- Using Buttons
- Using Check Boxes
- Using Combo Boxes and List Boxes
- Using Edit Boxes and Multi-Edit Boxes
- Using Frames
- Using Grids and HTML Tables
- Using Group Boxes
- Using HTML Objects
- Using Images
- Using Java Applets
- Using Lines
- Using Menus
- Using Radio Buttons
- Using Text
- Using ActiveX Controls
- Dynamically Manipulating Controls at Run Time
- Defining Colors for Your Form and Controls
- Assigning Help to Your Forms and Controls
- Assigning Tab Order to Controls
Which Control Should You Use?

Choose your control based on the task you wish the user to perform.

**If you want your end user to trigger an action at their convenience,** utilize the Button control or the Image control. For more information, see *Using Buttons* on page 165 or *Using Images* on page 223.

**If you want your end user to turn an option on and off,** utilize the Check Box control. For more information, see *Using Check Boxes* on page 167.

**If you want your end user to select one item from a list,** utilize the Radio button control, Combo Box control, or List Box control. For more information, see *Using Radio Buttons* on page 246 or *Using Combo Boxes and List Boxes* on page 172.

**If you want to display static data to an end user,** utilize the Text control. For more information, see *Using Text* on page 251.

**If you want to display data one data field at a time,** utilize the Edit Box or Multi-Edit Box control. For more information, see *Using Edit Boxes and Multi-Edit Boxes* on page 182.

**If you want to show multiple rows of data from a source,** use the Grid and HTML Table controls. For more information, see *Using Grids and HTML Tables* on page 193.

**If you want to show an object formatted as HTML,** use the HTML Object control. For more information, see *Using HTML Objects* on page 218.

**If you want to display a web link in a frame,** use the Frame control. For more information, see *Using Frames* on page 191.

**If you want to display a graphic,** use the Image control or the Background Image property. For more information, see *Using Images* on page 223 or *BackgroundImage Property* on page 270.

**If you want your end user to enter or edit data,** utilize the Edit Box or Multi-Edit Box control. For more information, see *Using Edit Boxes and Multi-Edit Boxes* on page 182.

**If you want to add a menu,** use the Menu control. For more information, see *Using Menus* on page 240.

**If you want to associate a set of options together and visually separate them from other things on a form,** use the Group Box control. For more information, see *Using Group Boxes* on page 217.

**If you want to draw a line,** use the Line control. For more information, see *Using Lines* on page 239.
Using Buttons

A button is a control that triggers an action when clicked.

For an example of how you can use a button, see the Maintain Getting Started manual.

Procedure: How to Place a Button on Your Form

1. Select the Button control in the Controls palette.
2. Draw a rectangle on your form approximately where you want your button to be and at approximately the size you want.
3. Type the text you want your button to have and press the Enter key. The text should be selected automatically when you created the button. If you want the text to appear on more than one line, press Shift+Enter between each line.
4. Giving your button a more meaningful name than Button is optional but recommended.
5. If necessary, readjust the size and placement of your button.

Note: If you click on the text of a button, you will be in text editing mode (when the text jumps to the top of the button and is selected). If you click near the edges of the button, you will be in move/resize mode. To change to a different mode, deselect the button and select it again.

6. Assign a Click action to your button using the Event Handler editor.

   Tip: You can open the Event Handler editor by double-clicking the button.

   For more information on assigning actions to events, see Defining Events and Event Handlers on page 125.

Changing Button Properties

When you select your button, you will see a list of button properties in the property sheet. Changing these properties will change what your button looks like and what it does at run time.

Many of the styling properties can also be changed using a Cascading Style Sheet. For more information on Cascading Style Sheets, see Using Cascading Style Sheets on page 113.

Choose your property based on the task you wish to perform:

If you want to change the text displayed in the button, use the Text Property. For more information, see the Text Property on page 292.
If you want to change the name of the button that identifies it to the procedure, use the (Name) Property. For more information, see the (Name) Property on page 267.

If you want to change the size or location of the button, use the Bottom Property, Left Property, Right Property, Top Property, or use a style sheet. For more information, see the Bottom Property on page 273, Left Property on page 286, Right Property on page 289, and Top Property on page 293. You can also move or resize the button directly in the form.

If you want to change the color of the button, use the BackColor Property to determine the color of the button, the ForeColor Property to determine the color of the text in the label, or use a style sheet. For more information, see the BackColorOver Property on page 270 and the ForeColor Property on page 279.

If you want to change the text font, use a style sheet or use the Font Property. For more information, see the Font Property on page 279.

If you want to make the button inactive or make it invisible, use the Enabled Property to determine whether the button is active or not. If the button is inactive, it will be dimmed out and nothing will happen when the end user clicks it. You can also use the Visible Property to determine whether the button is visible to the end user or use a style sheet. For more information, see the Enabled Property on page 278 and the Visible Property on page 294.

If you want to change what the cursor looks like when it is on top of the button, use the CursorPointer Property or use a style sheet. For more information, see the CursorPointer Property on page 276.

If you want to display a tooltip when the cursor is on the top of the button, use the ToolTipText Property. For more information, see the ToolTipText Property on page 293.

If you want to assign a Help topic to the button, use the Help Property. For more information, see Assigning Help to Your Forms and Controls on page 263.

If you want the end user to be able to tab to the button, use the Tabstop Property. For more information, see the Tabstop Property on page 291.

If you want to make the event handler for clicking the button occur when the end user presses the Enter key, use the DefaultButton Property. For more information, see the DefaultButton Property on page 277.

If you want to move the button to another layer, use the Layer Property. For more information, see Layering Controls on page 103.

If you want to control the display order of the button when more than one control is overlaid, use the ZIndex Property. For more information, see the ZIndex Property on page 294.
Using Check Boxes

Use check boxes in your forms to enable your end users to turn options on and off.

There are several ways you can use a check box in your form:

- If you want end users to view, edit, or add data to a data source, and one of the fields is a logical value (either yes or no), instead of displaying the value in an edit box, you can assign this value to a check box. This removes the possibility that the end user might insert incorrect data into the data source.

  For example, if you were creating a program that registered guests at your hotel, you might ask if they want a non-smoking room. Answering yes or no is as simple as selecting or clearing the check box.

- You can assign the check box state (either selected or cleared) to a variable, and then use the variable to determine an action to take.

- When an end user either selects or clears the check box, you can immediately perform some action. For example, a common action that might occur is that some other controls might become visible or active.

Procedure: How to Place a Check Box on Your Form

1. Select the Check box control in the Controls palette.
2. Draw a rectangle on your form approximately where you want your check box to be at approximately the size you want.
3. Select the text next to the actual box, type the label you want your check box to have, and press the Enter key.
4. Giving your check box a more meaningful name than CheckBoxn is optional, but recommended.
5. If necessary, readjust the size and placement of your check box.
6. Double-click the check box or select the Checked property to open the Set Check Box State dialog box.
7. To directly set the initial state for the check box, select As selected below, and then select either 0 - No (cleared) or 1 - Yes (selected).

   To assign the state of the check box to a variable, select From a variable. Then select a variable or data source stack column.
Note: You can create a new variable or data source stack by clicking New variable or New data source stack. For more information, see How to Create a Variable in a Procedure on page 68 or How to Create a Data Source Stack Explicitly Using the Stack Editor on page 73.

The variable or stack column should have a value of 1 or 0, if numeric.

8. Click OK.

Syntax:  How to Set the Value of a Check Box Dynamically

If you want to reset the value of a check box to its initial value, issue the following command

```
COMPUTE Formname.CheckBoxName.Checked = {0|1};
```

where:

- **Formname**
  - Is the name of the form the check box is placed on.

- **CheckBoxName**
  - Is the name of the check box.

- **0**
  - Clears the check box.

- **1**
  - Selects the check box.

Procedure:  How to Trigger an Action When an End User Selects a Check Box

1. Open the Event Handler editor.

2. Select the check box in the list of controls.

3. Select the Check event from the list of events.

4. Specify an event handler.
Reference: Set Check Box State Dialog Box

Use the Set Check Box State dialog box, shown in the image below, to determine whether the check box will be selected or cleared initially, or whether its value will be assigned to a variable or data source stack column.

This dialog box contains the following options:

**Set the checked state of the control**

Select *As selected below* to set the initial value directly.

Select *From a variable* to set the value from a variable.

**Checked**

This is available only if you selected *As selected below*.

Select *0 - No* to clear the check box initially (the default).

Select *1 - Yes* to select the check box initially.

**New variable**

This is available only if you select *From a variable*. This option opens the New Variable dialog box, where you can create a variable.
New data source stack

This is available only if you select *From a variable*. This option opens the Stack Editor, where you can create a stack.

List of data source stacks and variables in your procedure

This is available only if you select *From a variable*. This option contains a list of the existing stacks and variables in your procedure, as shown in the image below.

Either select a variable, or expand a data source stack and select a column.

![Set CheckBox State](image)

Data Sources - Current Area

This is available only if you select *From a variable*. This option lists the fields from the data sources used in this procedure.

Changing Check Box Properties

When you select your check box, you will see a list of check box properties in the property sheet. Changing these properties will change what your check box looks like and what it does at run time.

Many of the styling properties can also be changed using a Cascading Style Sheet. For more information on Cascading Style Sheets, see [*Using Cascading Style Sheets*](#) on page 113.

Choose your check box property based on the task you wish to perform:
If you want to change the label that identifies the check box to the end user, use the Text Property. For more information, see the Text Property on page 292.

If you want to change the name of the check box that identifies it to the procedure, use the (Name) Property. For more information, see the (Name) Property on page 267.

If you want to change the initial setting for the check box or assign the check box state to a variable, use the Check Box State dialog box. For more information, see Set Check Box State Dialog Box on page 169.

If you want to change the size or location of the check box, use the Bottom Property, Left Property, Right Property, Top Property, or use a style sheet. For more information, see the Bottom Property on page 273, Left Property on page 286, Right Property on page 289, and Top Property on page 293. You can also move or resize the check box directly in the form.

If you want to change the color of the check box, use the BackColor Property to determine the background color, the ForeColor Property to determine the color of the text in the label, or use a style sheet. For more information, see the BackColorOver Property on page 270 and the ForeColor Property on page 279.

If you want to change the label font, use a style sheet or use the Font Property. For more information, see the Font Property on page 279.

If you want to move the label to the other side of the check box, use the TextOnLeft Property. For more information, see the TextOnLeft Property on page 292.

If you want to make the check box inactive or make it invisible, use the Enabled Property to determine whether the check box is active or not. If the check box is inactive, it will be dimmed out and nothing will happen when the end user clicks it. You can also use the Visible Property to determine whether the check box is visible to the end user or use a style sheet. For more information, see the Enabled Property on page 278 and the Visible Property on page 294.

If you want to change what the cursor looks like when it is on top of the check box, use the CursorPointer Property or use a style sheet. For more information, see the CursorPointer Property on page 276.

If you want to display a tooltip when the cursor is on the top of the check box, use the ToolTipText Property. For more information, see the ToolTipText Property on page 293.

If you want to assign a Help topic to the check box, use the Help Property. For more information, see Assigning Help to Your Forms and Controls on page 263.

If you want the end user to be able to tab to the check box, use the Tabstop Property. For more information, see the Tabstop Property on page 291.
If you want to move the check box to another layer, use the Layer Property. For more information, see Layering Controls on page 103.

If you want to control the display order of the check box when more than one control is overlaid, use the ZIndex Property. For more information, see the ZIndex Property on page 294.

Using Combo Boxes and List Boxes

Combo boxes and list boxes enable end users to select a value from a list. List boxes display as much of the list as there is room in the box. If the list is larger than the box, WebFOCUS Maintain will display scroll bars to enable end users to view the other options. You can also enable list boxes so that the end user can select more than one option. Combo boxes are only one line long. To select an option, the end user must click the arrow at the right side of the box to see the list.

The contents of the list are determined dynamically at run time by one of the following:

- A list you supply at development time.
- The contents of a source stack.
- The list of valid values defined for a data source field in a data source description.

Procedure: How to Place a Combo Box or a List Box on Your Form

1. Select the Combo box control or the List box control in the Controls palette.
2. Draw a rectangle on your form approximately where you want your combo box or list box to be at approximately the size you want.
3. Giving your combo box or list box a more meaningful name than ComboBoxn or ListBoxn is optional, but recommended.
4. If necessary, readjust the size and placement of your combo box or list box.
5. Double-click the combo box or list box or select the ListItems property to open the List Source dialog box. Use this dialog box to determine the items in your list.
6. You have four options for defining the items in your combo box or list box:

   - Manually. Select As entered here and enter the items that will appear in your combo box or list box. For more information, see How to Enter Items Manually in the List Source Dialog Box on page 173.
From a data source stack column. Select From a variable and select the name of a data source stack column. You can create a data source stack by clicking New data source stack. For more information, see How to Create a Data Source Stack Explicitly Using the Stack Editor on page 73.

From a variable. Select From a variable and select the name of a variable. Only variables with the data type Stack of will appear here. You can create a variable by clicking New variable. For more information, see How to Create a Variable in a Procedure on page 68.

Note: You must create a variable with the data type, Stack of.

From the ACCEPT list of a data source field. Select From a variable and select the column in the list of variables. You will only see this option if any data source this procedure is using contains a field with an ACCEPT list.

7. Click OK.
8. Select the SelectedItem or SelectedItems property to open the Binding the Selection Result dialog box. Use this dialog box to bind the end users choice in the combo box or list box to a data source stack column or variable.

Caution: Do not bind SelectedItem or SelectedItems to a data source stack that already contains data. This will replace the contents of the data source stack with the result of the selection.

Note: Since end users can select multiple values in a list box, you must bind the value to a stack.

9. Select a data source stack column or variable. You can create a data source stack or variable by clicking New data source stack or New variable. For more information, see How to Create a Data Source Stack Explicitly Using the Stack Editor on page 73 or How to Create a Variable in a Procedure on page 68.

10. Determine whether to store the text or the values you entered in the List Source dialog box.
11. Click OK.
12. If you are placing a list box on your form and you want your end users to be able to select more than one item, set the MultiSelection property to 1 - Yes.

Procedure: How to Enter Items Manually in the List Source Dialog Box

To create an item in the List Source dialog box, do one of the following:
Click the New button to create the item and open the Enter a List Item dialog box where you can define your new item.

- Double-click the empty dim text block in the Text column to open the Enter a List Item dialog box.
- Select the item that you want placed previously to your new item and press the down arrow key.

To edit an item in the List Source dialog box, do one of the following:

- Double-click an item to open the Enter a List Item dialog box where you can edit the item.
- Select the item and press the F2 key.

**Tip:** To define the list, you can drag text from any application (for example, Notepad) on top of a combo box, list box, or group of radio buttons.

**Syntax:** How to Set the Value of a Combo Box or List Box Dynamically

If you want to set the value of a combo box or list box dynamically, issue the following command:

```
COMPUTE Formname.ControlName.ListItems.FocIndex = n;
```

where:

- **Formname**
  
  Is the name of the form the check box is placed on.

- **ControlName**
  
  Is the name of the combo box or list box.

- **ListItems**
  
  Is the name of an internal data source stack that contains the values for the combo box or list box.

You can also reset the values of all controls in a form to their initial values using the Winform Reset command. For more information, see *Manipulating Form Properties* on page 119.
**Syntax:** How to Preselect Items in a Multi-Select List Box by Value

You can preselect items in a list box with the MultiSelection property turned on using the following syntax:

```
Winform Show_Inactive Formname;Formname.ListBoxName.SelectedItems(1) = "value1";Formname.ListBoxName.SelectedItems(2) = "value2";
...
Formname.ListBoxName.SelectedItems(m) = "valuen";
```

where:

- **Formname**
  - Is the name of the form the check box is placed on.
- **ListBoxName**
  - Is the name of the list box.
- **SelectedItems**
  - Is the name of an internal data source stack that can contain an array of multiple preselected items. Like all stacks, it is a 1-based, not 0-based, array.

**Procedure:** How to Trigger an Action When Selecting From a Combo Box or List Box

1. Open the Event Handler editor.
2. Select the combo box or list box in the list of controls.
3. Select the Change event from the list of events.
4. Specify an event handler.
**Reference: List Source Dialog Box**

The List Source dialog box, shown in the following image, is used to determine the contents of your list box, radio button, or combo box.

![List Source Dialog Box](image)

This dialog box includes the following options:

**Insert list items**

Select *As entered here* to manually enter list items.

Select *From a variable* to retrieve the list from a data source stack column, variable with data type Stack of, or ACCEPT list for a data source field. This enables you to determine at run time which items should appear in the list. When you select this option, you will see a list of available Maintain variables in your procedure and data source fields with ACCEPT lists.

**Text**

Displays the prompt text for each item.

**Value**

Displays the values assigned to the prompt text for each item.
6. Developing and Using Controls

- **Add**: Adds a selected item to the list of fields by opening the Enter a List Item dialog box.
- **Delete**: Deletes a selected item from the list of items.
- **Up**: Moves a selected item up in the list of items.
- **Down**: Moves a selected item down in the list of items.

**New variable**

- Opens the New Variable dialog box, where you can create a variable with the data type, Stack of.

**New data source stack**

- Opens the Stack Editor, where you can create a stack.
**List of data source stacks and variables in your procedure**

Contains a list of the existing stacks and variables in your procedure, as shown in the image below. Either select a variable, or expand a data source stack and select a column.

**Data Sources - Fields with Accept List**

Lists the data sources used in this procedure with fields that have an ACCEPT list. You will not see this option if you do not have any such fields.
Reference: Enter a List Item Dialog Box

You can use the Enter a List Item dialog box, shown in the following image, to manually enter list items and assign values to the items you enter.

![Enter a List Item Dialog Box](image)

This dialog box contains the following elements:

Item text

Contains the list item displayed to the end user.

Assigned value (optional)

Contains the value generated if the end user selects this item. Later, you can store this value in the data source.

Reference: Binding the Selection Result Dialog Box

Use the Binding the Selection Result dialog box, shown in the following image, to bind your selection results to a variable or data source stack column as text, or as a value.
Caution: Do not bind your selection results to a data source stack that already contains data. This will replace the contents of the data source stack with the selection result.

This dialog box contains the following elements:

**Bind the selection result to this variable**
- Contains whatever you selected below.

**Clear**
- Removes your current selection if you do not want to bind the selection result.

**New variable**
- Opens the New Variable dialog box, where you can create a variable.

**New data source stack**
- Opens the Stack Editor, where you can create a stack.

**List of data source stacks and variables in your procedure**
- Contains a list of the existing stacks and variables in your procedure.
  - Either select a variable, or expand a data source stack and select a column.
Data Sources - Current Area

Lists the fields from the data sources used in this procedure.

Send the result as

Select Text to save the text value you entered in the List Source dialog box.

Select Value to save the value you entered in the List Source dialog box.

Changing Combo Box or List Box Properties

When you select your combo box or list box, you will see a list of combo box or list box properties in the property sheet. Changing these properties will change what your edit box or multi-edit box looks like and what it does at run time.

Many of the styling properties can also be changed using a Cascading Style Sheet. For more information on Cascading Style Sheets, see Using Cascading Style Sheets on page 113.

Choose your combo box property or list box property based on the task you wish to perform:

If you want to change the contents of the combo box or list box, use the ListItems Property to open the List Source dialog box. For more information, see the List Source Dialog Box on page 176.

If you want to change the name of the combo box or list box that identifies it to the procedure, use the (Name) Property. For more information, see the (Name) Property on page 267.

If you want to assign the value selected by the end user to another variable, use the SelectedItem or SelectedItems Property to open the Binding the Selection Result dialog box. For more information, see Binding the Selection Result Dialog Box on page 179.

If you want to enable an end user to select more than one item in a list box, use the MultiSelection Property. This property does not apply to combo boxes. For more information, see the MultiSelection Property on page 286.

If you want to change the size or location of the combo box or list box, use the Bottom Property, Left Property, Right Property, Top Property, or use a style sheet. For more information, see the Bottom Property on page 273, Left Property on page 286, Right Property on page 289, and Top Property on page 293. You can also move or resize the combo box or list box directly in the form.

If you want to change the color of the combo box or list box, use the BackColor Property to determine the color of the box, the ForeColor Property to determine the color of the text in the box, or use a style sheet. For more information, see the BackColorOver Property on page 270 and the ForeColor Property on page 279.
If you want to change the font in the combo box or list box, use a style sheet or use the Font Property. For more information, see the Font Property on page 279.

If you want to make the combo box or list box inactive or make it invisible, use the Enabled Property to determine whether the combo box or list box is active or not. If the combo box or list box is inactive, it will be dimmed out and nothing will happen when the end user clicks it. You can also use the Visible Property to determine whether the combo box or list box is visible to the end user or use a style sheet. For more information, see the Enabled Property on page 278 and the Visible Property on page 294.

If you want to display a tooltip when the cursor is on the top of the combo box or list box, use the ToolTipText Property. For more information, see the ToolTipText Property on page 293.

If you want to assign a Help topic to the combo box or list box, use the Help Property. For more information, see Assigning Help to Your Forms and Controls on page 263.

If you want the end user to be able to tab to the combo box or list box, use the Tabstop Property. For more information, see the Tabstop Property on page 291.

If you want to move the combo box or list box to another layer, use the Layer Property. For more information, see Layering Controls on page 103.

If you want to control the display order of the combo box or list box when more than one control is overlaid, use the ZIndex Property. For more information, see the ZIndex Property on page 294.

Using Edit Boxes and Multi-Edit Boxes

Edit boxes and multi-edit boxes enable end users to view, enter, and edit data for a stack or user-defined field. Edit boxes allow only single lines of text, while multi-edit boxes allow multiple lines.

Procedure: How to Place an Edit Box or Multi-Edit Box on Your Form

1. Select the Edit box control or Multi-edit box control in the Controls palette.
   
   Draw a rectangle on your form approximately where you want your edit box or multi-edit box to be at approximately the size you want.

2. Giving your edit box or multi-edit box a more meaningful name than EditBoxn or MultiEditBoxn is optional, but recommended.

3. If necessary, readjust the size and placement of your check box.
4. Double-click the edit box or multi-edit box or select the Text property to open the Insert Text dialog box. You will use the dialog box to determine the data source for your edit box or multi-edit box.

5. To directly set the initial text for the edit box or multi-edit box, select As entered below, and then enter the text in the box.

   To assign the contents of the edit box or multi-edit box to a variable, select From a variable, and then select a variable or data source stack column. You can create a new variable or data source stack by clicking New variable or New data source stack. For more information, see How to Create a Variable in a Procedure on page 68.

6. Click OK.

**Note:**

- If you bind a variable to an edit box, the end user is restricted to the number of characters defined by the print width of the type of the variable. A multi-edit box cannot restrict the number of characters entered.

- Typically, you assign multi-edit boxes to a variable with the data type A0 and then use the SUBSTR built-in function to break the variable contents into A256 fields for storage in the data source. For more information on SUBSTR, see the Using Functions manual.
Reference: Insert Text Dialog Box

Use the Insert dialog box, shown in the following image, to populate edit and multi-edit boxes with data.

![Insert Text Dialog Box]

This dialog box contains the following elements:

**Insert text into the control**

- Select *As entered below* to set the text directly.
- Select *From a variable* to set the value from a variable.

**Maximum length (-1 for no limit)**

This is available only if you select *As entered below*. This option enters the maximum number of characters this edit box or multi-edit box can contain.

**New variable**

This is available only if you select *From a variable*. This option opens the New Variable dialog box, where you can create a variable.

**New data source stack**

This is available only if you select *From a variable*. This option opens the Stack Editor, where you can create a stack.
List of data source stacks and variables in your procedure

This is available only if you select *From a variable*. This option contains a list of the existing stacks, variables, and database fields in your procedure, as shown in the image below.

Either select a variable, or expand a data source stack or database and select a column.

Data Sources - Current Area

This is available only if you select *From a variable*. This option lists the fields from the data sources used in this procedure.

**Note:** We recommend that you do not use the Current Area. Data source stacks are a superior way of accessing and manipulating data source values, and they function more intuitively than the Current Area.

Changing Edit or Multi-Edit Box Properties

When you select your edit box or multi-edit box, you will see a list of edit box or multi-edit box properties in the property sheet. Changing these properties will change what your edit box or multi-edit box looks like and what it does at run time.

Many of the styling properties can also be changed using a Cascading Style Sheet. For more information on Cascading Style Sheets, see *Using Cascading Style Sheets* on page 113.
Choose your edit box property or multi-edit box property based on the task you wish to perform:

**If you want to change the contents of the edit box of multi-edit box**, use the Text Property to open the Insert Text dialog box. For more information, see the *Insert Text Dialog Box* on page 184.

**If you want to change the name of the edit box or multi-edit box that identifies it to the procedure**, use the *(Name) Property*. For more information, see the *(Name) Property* on page 267.

**If you want to change the size or location of the edit box or multi-edit box**, use the Bottom Property, Left Property, Right Property, Top Property, or use a style sheet. For more information, see the *Bottom Property* on page 273, *Left Property* on page 286, *Right Property* on page 289, and *Top Property* on page 293. You can also move or resize the edit box or multi-edit box directly in the form.

**If you want to change the color of the edit box or multi-edit box**, use the BackColor Property to determine the color of the box, the ForeColor Property to determine the color of the text in the box, or use a style sheet. For more information, see the *BackColorOver Property* on page 270 and the *ForeColor Property* on page 279.

**If you want to change the font in the edit box or multi-edit box**, use a style sheet or use the Font Property. For more information, see the *Font Property* on page 279.

**If you want to change the case of text entered in the edit box or multi-edit box before being returned to the bound variable**, use the CaseStyle Property. For more information, see the *CaseStyle Property* on page 274.

**If you want to display only asterisks, instead of actual characters when an end user enters information into an edit box**, use the Password Property. This property does not apply to multi-edit boxes. For more information, see the *Password Property* on page 288.

**If you want to make the contents of the edit box or multi-edit box read-only**, use the ReadOnly Property. You can accomplish this by disabling the Enable property, but this method does not dim the control. For more information, see the *ReadOnly Property* on page 289.

**If you want to remove the border or add a border to the edit box or multi-edit box**, use the Border Property on page 271.

**If you want to make the edit box or multi-edit box inactive or make it invisible**, use the Enabled Property to determine whether the edit box or multi-edit box is active or not. If the edit box or multi-edit box is inactive, it will be dimmed out and nothing will happen when the end user clicks it. You can also use the Visible Property to determine whether the edit box or multi-edit box is visible to the end user or use a style sheet. For more information, see the *Enabled Property* on page 278 and the *Visible Property* on page 294.
If you want to change what the cursor looks like when it is on top of the edit box or multi-edit box, use the CursorPointer Property or use a style sheet. For more information, see CursorPointer Property on page 276.

If you want to display a tooltip when the cursor is on the top of the edit box or multi-edit box, use the ToolTipText Property. For more information, see the ToolTipText Property on page 293.

If you want to assign a Help topic to the edit box or multi-edit box, use the Help Property. For more information, see Assigning Help to Your Forms and Controls on page 263.

If you want the end user to be able to tab to the edit box or multi-edit box, use the Tabstop Property. For more information, see the Tabstop Property on page 291.

If you want to move the edit box or multi-edit box to another layer, use the Layer Property. For more information, see Layering Controls on page 103.

If you want to enable automatic validation of data formats at run time, use the IBIValidation Property on page 283.

If you want to control the display order of the combo box or list box when more than one control is overlaid, use the ZIndex Property. For more information, see the ZIndex Property on page 294.

Using Prompted Edit Boxes

A prompted edit box consists of two controls grouped together, an edit box and a text control describing the edit box, as shown in the following image.

WebFOCUS Maintain generates prompted edit boxes for you automatically during the following situations:

- When you drag a data source stack from the Project Explorer into the Form Editor. WebFOCUS Maintain displays the Select Stack Columns dialog box where you specify which columns in the data source stack you want to create prompted edit boxes for.
- When you drag a data source stack column from the Project Explorer into the Form Editor.
- When you drag a variable from the Project Explorer into the Form Editor.
- When you drag a segment from a data source in the Project Explorer into the Form Editor. WebFOCUS Maintain displays the Select Segment Fields dialog box to prompt you for the fields for which you want to create prompted edit boxes and what the data source will be.
When you drag a data source field from the Project Explorer into the Form Editor. The data source for the edit box will be the data source field in the Current Area.

**Note:** We recommend that you do not use this method for generating prompted edit fields. Data source stacks are a superior way of accessing and manipulating data source values, and they function more intuitively than the Current Area.

If you want to change the properties for either the edit box or the text control separately, select the prompted edit box. Then select either the edit box or the text control in the dropdown list of grouped controls in the property sheet.

For more information on grouping controls, see *Grouping Controls* on page 101.

WebFOCUS Maintain includes a special alignment command for aligning prompted edit boxes along the left sides of the edit boxes. For more information, see *How to Align Prompted Edit Boxes Along the Left Sides of the Edit Boxes* on page 100.

**Reference:**  **Select Stack Columns Dialog Box**

When you drag a data source stack from the Project Explorer into the Form Editor, WebFOCUS Maintain displays the Select Stack Columns dialog box, shown in the following image. Use this dialog box to determine which columns in the data source stack for which to create prompted edit boxes.

![Select Stack Columns Dialog Box](image)

This dialog box has the following options:

**Select columns to display from this stack**

Select the columns in this data source stack that you want to display on your form. Clear the columns you do not want to display.
Select all
Selects all the columns in the data source stack.

Clear all
Clears all the columns in the data source stack.

Reference: Select Segment Fields Dialog Box
When you drag a segment from a data source in the Project Explorer into the Form Editor, WebFOCUS Maintain displays the Select Segment Fields dialog box, shown in the following image. Use this dialog box to determine which fields in the data source segment WebFOCUS Maintain should create prompted edit boxes for and what their data source should be.

When you finish specifying information in this dialog box and click OK, WebFOCUS Maintain will do the following:

- Create prompted edit boxes for the data source fields you select here to the Form Editor.
- If you created a new data source stack, add the data source stack to the list of data source stacks in the procedure that the form is in.
Add the data source you specified to the procedure, shown in the following image, if it is not already part of it.

This dialog box has the following options:

**Select fields to display from this segment**
Select the fields in this data source segment that you want to display on your form. Clear the fields you do not want to display.

**Select all**
Selects all the fields in the segment.

**Clear all**
Clears all the fields in the segment.

**Bind fields to**
Select *Current Area* to specify that the data source for the edit fields will be the Current Area.
Select `New stack` to specify that the data source for the edit fields will be the data source stack you specify here. WebFOCUS Maintain suggests a name for this data source stack, but you can type another name here.

**Note:**
- We recommend that you do not use the Current Area. Data source stacks are a superior way of accessing and manipulating data source values, and they function more intuitively than the Current Area.
- You cannot enter the name of a data source stack that already exists. If you want to use an existing data source stack as the data source for your prompted edit fields, click `Cancel` and drag the data source stack you want from the Project Explorer into the Form Editor.

**Select a single path to include in your stack**

Enables you to select all of the fields in a single segment of a hierarchical database. If you have a flat file or relational database this enables you to select all of the fields.

**Using Frames**

The Frame control enables you to launch a web link in a frame on your form instead of a separate window.

**Procedure: How to Launch a Web Link in a Frame**

1. Select the `Frame` control in the Controls palette.
2. Draw a rectangle on your form approximately where you want your frame, at approximately the size you want.
3. Giving your frame a more meaningful name than `Frame` is optional, but recommended.
4. If necessary, readjust the size and placement of your frame.
5. Double-click the frame, or select the Source property to open the URL Link dialog box.
6. If you have an existing web link that you would like to use, select its name from the `Use the URL link from this resource` list. Otherwise, click `New` to create a new web link. For more information, see *How to Create a Web Link* on page 144.
7. Click `OK`. 
**Reference:**  **URL Link Dialog Box**

Use this dialog box to specify the URL link for a frame on your form.

![URL Link Dialog Box](image)

This dialog box has the following fields:

**Use the URL link from this resource**

Enables you to select one of the web links you have already defined in your project.

**New**

Opens the Link Editor dialog box, where you can define a new web link.

**Edit**

If you select a web link from the list above, clicking here opens the Link Editor dialog box, where you can edit an existing web link in your project.

**Changing Frame Properties**

When you select your frame, you will see a list of frame properties in the property sheet. Changing these properties will change what your frame looks like and how it behaves at run time.

Many of the styling properties can also be changed using a Cascading Style Sheet. For more information on Cascading Style Sheets, see *Using Cascading Style Sheets* on page 113.

Choose the property based on the task you wish to perform:

**If you want to change the contents of the frame,** use the Source Property to open the URL Link dialog box. For more information, see the **URL Link Dialog Box** on page 192.

**If you want to change the name of the frame that identifies it to the procedure,** use the (Name) Property. For more information, see the **(Name) Property** on page 267.

**If you want to change the size or location of the frame,** use the Bottom Property, Left Property, Right Property, Top Property, or use a style sheet. For more information, see the **Bottom Property** on page 273, **Left Property** on page 286, **Right Property** on page 289, and **Top Property** on page 293. You can also move or resize the frame directly in the form.
If you want to add scroll bars to the frame, scrolling is automatically provided by the browser at run time if the content exceeds the size of the frame control.

If you want to add a border to the frame, use the Border Property or use a style sheet. For more information, see Border Property on page 271.

If you want to make the frame invisible, use the Visible Property or use a style sheet. For more information, see Visible Property on page 294.

If you want to display a tooltip when the cursor is on the top of the frame, use the ToolTipText Property. For more information, see ToolTipText Property on page 293.

If you want to move the frame to another layer, use the Layer Property. For more information, see Layering Controls on page 103.

If you want to control the display order of the frame when more than one control is overlaid, use the ZIndex Property. For more information, see the ZIndex Property on page 294.

Using Grids and HTML Tables

The Grid controls and the HTML Table control display the contents of a data source stack to end users. The Grids differ from the HTML Table in that they enable you to directly change data in a stack.

There are two different read/write grids available: the ActiveX Grid and the JavaScript Grid.

The ActiveX Grid control has been available since WebFOCUS Maintain 4. It supports JavaScript and VBScript.

Installation of WebFOCUS Developer Studio will automatically register the control on your machine to enable it for local deployment.

For web deployment, the first time an application containing the ActiveX Grid control is run, it requires a one-time download of the signed ActiveX control for registration on the client (except if WebFOCUS Developer Studio is also installed on the client). The browser security settings under Internet Options should allow a prompt or enable for downloading signed ActiveX controls.

Some browsers, such as Firefox®, do not support ActiveX.

The JS Grid is available for new development as of WebFOCUS Maintain Release 7.7. It is not an ActiveX control and does not require anything to be registered on the client. It can be run in non-IE browsers, such as Firefox.

Note: VBscript is not supported with the JS Grid.
The ActiveX Grid and the JS Grid share the same dialogs and most settings within the MDE, including Properties and Events. The exceptions are noted. Functions must be coded differently and are outlined in separate sections.

For an example of using the original ActiveX grid, see the WebFOCUS Maintain Advanced Tutorial in the Maintain Getting Started manual. For an example of using HTML tables, see the WebFOCUS Maintain Basic Tutorial in Maintain Getting Started manual and Using an HTML Table to Display the Results of a WebFOCUS Report on page 318.

Procedure: How to Place a Read/Write Grid or an HTML Table on a Form

1. Select the ActiveX Grid control, the JS Grid control, or the HTML Table control in the Controls palette.
2. Draw a rectangle on your form approximately where you want to place your read/write grid or HTML table, at approximately the size you want.
3. WebFOCUS Maintain automatically opens the Control Columns dialog box. You can also open this dialog box by double-clicking the read/write grid or with the Columns property. Select a data source stack from the Stack drop-down list. You can create a data source stack by clicking New. For more information, see How to Create a Data Source Stack Explicitly Using the Stack Editor on page 73.
4. Select the columns in the data source stack that you want to display and click the right arrow button. WebFOCUS Maintain adds these columns to the Table Columns box.
   To remove a column from the Table Columns box, select it and click the left arrow button.
5. If you want to remove row numbers, clear the Include a row number column check box.
6. If you want to change the order of the columns in the Column Selection box, select them and use the up and down arrows.
7. To sort your columns by title, column, or width, click the corresponding button (Header Title, Column, or Width) in the Table Columns box.
8. To format a column, double-click in the Table Columns box to open the Grid Column Properties or Table Column dialog box.
9. Click OK to leave the Control Columns dialog box.
10. Giving your grid or HTML Table a more meaningful name than gridn or HTMLTablen is optional, but recommended.
11. If necessary, readjust the size and placement of your grid or HTML Table.
Tip: If you want to add a column to your HTML Table from the data source stack used by the table, drag it from the Project Explorer to the HTML Table. WebFOCUS Maintain will add it to the end.

Reference: Control Columns Dialog Box

Use the Control Columns dialog box, shown in the following image, to populate a grid or HTML Table with data.

This dialog box includes the following options:

**Stack**

Displays the data source stacks available in this procedure.

**New**

Creates a data source stack with the Stack Editor.

**List of columns in selected data source stack**

Contains the columns in the data source stack you selected in the Stack list.
Includes the column you selected in the list of data source stack columns in the Table columns box.

Removes the column you selected in the Table columns box.

**Table columns**

Lists the columns that the HTML Table will display.

**Header Title**

Displays each of the column headings, and sorts the table by the header title when clicked.

**Header Height**

The default height for the header can be changed to accommodate additional lines in the Header Title. The default value for the Header Height is 23.

**Note:** Changing this value for a column adjusts the height for the entire header line.

Applies to HTMLTable and JS Grid.

**Column**

Displays the name of the column in the data source stack, and sorts the table by its column name when clicked.

**Width**

Displays the number of characters in a column, and sorts the table by width when clicked. Data values will automatically wrap when the length exceeds the width of the column.

**Include a row number column**

Displays row numbers in the HTML Table when checked.

Moves a selected column up in the list of columns.

Moves a selected column down in the list of columns.
Reference: Table Column/Grid Column Properties Dialog Box

Use the Grid Column Properties/Table Column dialog box, shown in the following image, to format columns in your grid or HTML Table.

This dialog box contains the following elements:

Stack column

Displays the column name.

Header title

Contains the title of the column that will be displayed at run time. By default, this is the column name, but you can type something else here.
**Header height**

The default height for the header can be changed to accommodate additional lines in the Header Title. The default value for the Header height is 23.

**Note:** Changing this value for a column adjusts the height for the entire header line. This applies to HTMLTable and JS Grid.

**Width in characters**

Contains the width of the column. You can type a new value here.

**Justification**

Enables you to left-justify, center, or right-justify the contents of a column.

**Content Type**

For HTML Tables only. Specifies the content of your column as text, HTML, or the path to an image. If you specify the content of your column as text, your stack column will be displayed in the table. If you specify the content of your column as HTML, WebFOCUS Maintain assumes that you have HTML code in your stack column. Therefore, it interprets the column as HTML. If you specify the path to an image, WebFOCUS Maintain assumes that the stack column contains the name and location of an image. Note that if the path contains backslashes, you must use double backslashes (for example, C:\ibi_img\graphic.gif).

**Links**

For HTML Tables only. Specifies whether your table will have drill-down links from the column body or heading. For more information, see *Creating Links in HTML Tables* on page 204.

**Read Only**

For Grids only. Prohibits users from writing to the read/write grid when checked.

**Font**

Changes the font, font style, and size of the text in your column or header.

**Text Color**

Changes the color of the text in your column or header. For more information on text color, see *Defining Colors for Your Form and Controls* on page 260.

**Background Color**

Changes the background color of your column or header. For more information on background color, see *Defining Colors for Your Form and Controls* on page 260.
Events Available in a Read/Write Grid

The following section lists the events available for the ActiveX and JS read/write grids.

**Note:** Maintain and IWC triggers are not supported for the following events: Focus, OnEditStart, OnEditFinish, OnColSized, OnColChange, OnCanViewMove. These events must be called using JavaScript (JS grid, ActiveX grid) or VB Script (ActiveX grid).

**Blur**

Occurs when the cursor or focus is moved to another cell or object. This event is called by On Entry in the V3 grid.

**Focus**

Occurs when the cell is highlighted. Maintain triggers and IWC triggers are not supported for this event. This event is called by On Exit in the V3 grid.

**OnCanViewMove**

Occurs any time a new column or row comes into view by each up, down, left or right arrow key and also by the scroll bar.

It will trigger for each row or cell, so the scroll bar will cause it to trigger repeatedly. Maintain triggers and IWC triggers are not supported for this event. This event is called by ViewPortChanged in the V3 grid.

**OnCBDClicked**

Occurs when the end user double-clicks the corner button.

**OnCBLClicked**

Occurs when the end user left mouse clicks on the corner button.

**OnCBRClicked**

Occurs when the end user right-clicks on the corner button.

**OnCellChange**

Occurs when a cell changes focus.

**OnCellChanged**

Occurs when the cursor moves to another control.

**OnCharDown**

Occurs when the end user presses a character on the keyboard.
**OnColChange**

Occurs when a column changes focus. Maintain triggers and IWC triggers are not supported for this event.

**OnColSized**

This notification can be used to assign any given columns a specific width. If you are creating an application such as a calendar, you must assign each column a specific width. Maintain triggers and IWC triggers are not supported for this event.

**OnEditFinish**

This notification enables you to cancel any edits which the user has performed on the cell. Maintain triggers and IWC triggers are not supported for this event.

**OnEditStart**

Occurs when an edit is about to start. You can edit a cell by calling the StartEdit function. For example, a message box can be used to inform users when they are about to edit a cell. Maintain triggers and IWC triggers are not supported for this event.

**OnHitBottom**

Occurs if the bottom row of the grid has been reached. By processing this notification, new rows can be added to the grid dynamically. If the grid tries to go past its current bottom, it first asks the default data source to check and see if more records exist (up to numrows+rowspast).

**OnKeyDown**

Occurs when the end user presses a key on the keyboard. This event encompasses the following events from V3: Page Up, Page Down, Home, End, Up Arrow, and Down Arrow.

**OnLClicked**

Occurs when the end user left mouse clicks on a given cell.

**OnRClicked**

Occurs when the end user right-clicks on a given cell.

**OnRowChange**

Occurs when the end user changes the focus from Row A to Row B.

**OnRowSized**

Occurs when the end user changes the row size.
OnRowSizing

You can use this notification to set a default size for all of the rows. After this is set, the end user cannot change the size of the rows.

OnSHDClicked

Occurs when the end user double-clicks the side heading.

OnSHLClicked

Occurs when the end user left-clicks the side heading.

OnSHRClicked

Occurs when the end user right-clicks on the side heading.

OnTHDClicked

Occurs when the end user double-clicks event the Top heading. Remember the top heading has a value of -1, and if there are multi-top headings, then each heading has a value which decrements by -1.

OnTHLClicked

Occurs when the end user left-clicks the Top Heading.

OnTHRClicked

Occurs when the end user right-clicks the Top heading.

Note: The following parameters are reserved for internal use and are not to be reset by the developer: processed, updn, and cancelflag.

Changing the Properties of a Grid or HTML Table

When you select your grid or HTML Table, you will see a list of grid properties in the property sheet. Changing these properties will change what your grid or HTML Table looks like and how it behaves at run time.

Many of the styling properties can also be changed using a Cascading Style Sheet. For more information on Cascading Style Sheets, see Using Cascading Style Sheets on page 113.

Choose your property based on the task you wish to perform:

If you want to change the contents of the grid or HTML Table, use the Columns Property to open the Control Columns dialog box. For more information, see Control Columns Dialog Box on page 195.
If you want to change the name of the grid or HTML Table that identifies it to the procedure, use the (Name) Property. For more information, see the (Name) Property on page 267.

If you want to change the size or location of the grid, use the Bottom Property, Left Property, Right Property, Top Property, or use a style sheet. For more information, see the Bottom Property on page 273, Left Property on page 286, Right Property on page 289, and Top Property on page 293. You can also move or resize the grid directly in the form.

If you want to change the size or location of the HTML Table, use the Overflow Property set to Clip or Scroll. The Clip setting allows you to restrict the length of the HTML Table. Only the Scroll setting allows the entire HTML Table, including the width, to be resized. Vertical and horizontal scroll bars are automatically provided when the data exceeds the control size. Use the Bottom, Left, Right, or Top properties, if they are regulated by the Overflow setting. You can also move the location of the HTML Table directly on the form.

If you want to omit the header, use a style sheet or use the Headers Property on page 282.

If you want to change the color of the grid or HTML Table, the BackColor and ForeColor properties have the precedence in the following order:

1. Use the Columns Property to open the Control Columns dialog box and then double-click a column to open the Table Column/Grid Column Properties Dialog Box on page 197.

2. Use the HeaderForeColor Property to determine the color of the text in the header. For more information, see HeaderForeColor Property on page 282.

3. Use the HeaderBackColor Property to determine the color of the background of the header. For more information, see HeaderBackColor Property on page 281.

4. Use the AlternateRowColor Property to turn every other row a different color. For more information, see AlternateRowColor Property on page 268.

5. Use the ForeColor Property to determine the color of the text. For more information, see ForeColor Property on page 279.

6. Use the BackColor Property to determine the background color. For more information, see BackColor Property on page 269.

7. Use the column header color.

8. Use the column back or fore color, which overrides the column color, but not the control header color.

9. Use the control header color.

10. Use the control back or fore color.

For an HTML Table only, you can use a style sheet.
If you want to control which columns remain stationary while scrolling the grid, use the FixedColumns Property. For more information, see FixedColumns Property on page 278.

If you want to change the text of the font, the Font Property has precedence in the following order:

1. Use the Columns Property to open the Control Columns dialog box and then double-click a column to open the Table Column/Grid Column Properties dialog box. For more information, see Table Column/Grid Column Properties Dialog Box on page 197.
2. Use the Font Property and HeaderFont Property. For more information, see Font Property on page 279 and the HeaderFont Property on page 281.
3. Use the column header font.
4. Use the column back font, which overrides the column font but not the control header font.
5. Use the control header font.
6. Use the control back font.

For a HTML Table only, you can use a style sheet.

If you want to add a border to the HTML Table, use the Border Property on page 271 to determine whether you have a border and what type of border it is. The BorderColor Property allows you to determine the color of the border. The BorderWidth Property allows you to determine the width of the border. You may need to increase the BorderWidth and change BorderColor in order to take full advantage of some of the border types. You can also use a style sheet for HTML Table only.

If you want to display lines between columns and rows, use a style sheet for HTML Table only or use the GridLines Property. For more information, see GridLines Property on page 280.

If you want to make the grid or HTML Table inactive or make it invisible, use the Enabled Property to determine whether the button is active or not. If the button is inactive, it will be dimmed out and nothing will happen when the end user clicks it. You can also use the Visible Property to determine whether the button is visible to the end user or use a style sheet. For more information, see the Enabled Property and the Visible Property.

If you want to change what the cursor looks like when it is on top of the grid or HTML Table, use the CursorPointer Property or use a style sheet. For more information, see the CursorPointer Property on page 276.

If you want to display a tooltip when the cursor is on the top of the grid or HTML Table, use the ToolTipText Property. For more information, see the ToolTipText Property on page 293.

If you want to assign a Help topic to the grid or HTML Table, use the Help Property. For more information, see Assigning Help to Your Forms and Controls on page 263.
If you want to move the grid or HTML Table to another layer, use the Layer Property. For more information, see Layering Controls on page 103.

If you want to control the display order of the HTML Table or JS Grid when more than one control is overlaid, use the ZIndex Property. This cannot be used with the ActiveX grid control. For more information, see the ZIndex Property on page 294.

Creating Links in HTML Tables

A link in an HTML Table enables the end user to click a cell to drill down to other tables, forms, reports, or webpages.

For an example of this process, see the Maintain Getting Started manual.

Procedure: How to Create Links in an HTML Table Column

1. Select the HTML Table and use the Columns property to open the Control Columns dialog box.
2. Double-click a column that you want to be clickable to open the Table Column dialog box.
3. In the Links list, select Header only if you want the header for that column to be clickable. Select Body only if you want the body of the column to be clickable. Select Body and Header if you want both the header and body to be clickable.
4. Click OK to close the Table Column dialog box and return to the Control Columns dialog box.
5. Repeat steps 2 through 4 for any additional columns you want to be clickable.
6. Click OK to close the Control Columns dialog box.

When you deploy and run your application, you will see that the areas you designated as clickable, will be blue and underlined.

7. Open the Event Handler editor for the HTML Table and select the ClickLink event.
8. Enter the code for the action to be performed when an end user clicks one of the links in the HTML Table.

   If your action is a Maintain function, you can use special syntax to determine the row number, column number, or value of the cell that is clicked. For more information, see How to Determine Row Number, Column Number, or Value of the Cell When an HTML Table Is Clicked on page 205.

   If your action is a JavaScript or VBScript function, you can use special syntax to determine the row number, column number, or value of the cell that the end user clicked. For more information, see Defining Events and Event Handlers on page 125.
Note: Because clicking on a row in an HTML Table does not result in a selection (unlike clicking an item in a list or combo box), WebFOCUS Maintain does not set the FocIndex of the underlying data source stack to the number of the current row (as it does for list box or combo box selections). To set FocIndex to the last clicked row, see How to Reset FocIndex in a Data Source Stack for an HTML Table on page 205.

Syntax: How to Determine Row Number, Column Number, or Value of the Cell When an HTML Table Is Clicked

If you use a Maintain function to handle the ClickLink event in an HTML Table, then you can use special syntax in your function to determine what part of the HTML Table the end user clicked.

If you want to determine the row number, use:

```
formname.tablename.ClickRow
```

The header row returns 0, and the first data row returns 1.

If you want to determine the column number, use:

```
formname.tablename.ClickColumn
```

If you want to indicate the value of the cell, use:

```
formname.tablename.Value
```

Syntax: How to Reset FocIndex in a Data Source Stack for an HTML Table

When the end user clicks a row in an HTML Table, use the following syntax to reset the value of FocIndex (the number of the current row) for the data source stack associated with that table:

```
COMPUTE stack.FocIndex = form.HTMLTable.ClickRow;
```

This is useful if you want to perform some operation on that particular row of the data source stack (display it or change it).

Using Functions in the Read/Write Grids

Within the read/write grids, you can use script functions to alter the properties of the grid. Functions for the ActiveX Grid can be written using either VBScript or JavaScript. Functions for the JS Grid can be written in JavaScript only. For a complete list of functions available for each grid, see Property Functions in the ActiveX Read/Write Grid on page 206 and Property Functions in the Read/Write JS Grid on page 212. Examples are given for the most commonly used functions.
## Property Functions in the ActiveX Read/Write Grid

<table>
<thead>
<tr>
<th>Name of Function</th>
<th>VBScript</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddFont</td>
<td><code>fontid=Formn.Gridn.AddFont(fontid, 0, 1)</code></td>
<td>Text in the cell indicated by column and row number will appear in the font indicated by <code>fontid</code>.</td>
</tr>
<tr>
<td>QuickSetFont</td>
<td><code>Formn.Gridn.QuickSetFont c,r,fontid</code></td>
<td>Sets font for specified cells.</td>
</tr>
<tr>
<td>RedrawAll</td>
<td><code>Formn.Gridn.RedrawAll</code></td>
<td>Updates grid when you have made changes.</td>
</tr>
<tr>
<td>AddTab</td>
<td><code>Formn.Gridn.AddTab &quot;text&quot;, n</code></td>
<td>When the tab is clicked the sheet will display the text in cell <code>n</code>.</td>
</tr>
<tr>
<td>CellGetTextColor</td>
<td><code>color = Formn.Gridn.CellGetTextColor(r g b)</code></td>
<td>Specifies the color of the text in a cell. Values for <code>r</code>, <code>g</code>, and <code>b</code> specify the desired intensity of red, green, and blue. The values are on a scale of 0 to 255.</td>
</tr>
<tr>
<td>QuickSetTextColor</td>
<td><code>Formn.Gridn.QuickSetTextColor 0, 0, RGB(r,g,b)</code></td>
<td>Sets the text color of the specified cell.</td>
</tr>
<tr>
<td>QuickSetBackColor</td>
<td><code>Formn.Gridn.QuickSetBackColor 0, 0, RGB(r,g,b)</code></td>
<td>Sets the background color of the specified cell.</td>
</tr>
<tr>
<td>GetCurrentColumn</td>
<td><code>column=Formn.Gridn.GetCurrentColumn</code></td>
<td>Identifies the column that is clicked on or selected by user.</td>
</tr>
<tr>
<td>GetCurrentRow</td>
<td><code>row=Formn.Gridn.GetCurrentRow</code></td>
<td>Identifies the row that is clicked on or selected by user.</td>
</tr>
<tr>
<td>Name of Function</td>
<td>VBScript</td>
<td>Result</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>QuickSetMask</td>
<td><code>Formn.Gridn.QuickSetMask 0, 0, &quot;###&quot;</code></td>
<td>Selectively includes or excludes certain values from a field. For more information on how to use Mask options, see <em>How to Use the Mask Function With the ActiveX Grid</em> on page 207.</td>
</tr>
<tr>
<td>QuickSetAlignment</td>
<td><code>Formn.Gridn.QuickSetAlignment 0, 0, 2</code></td>
<td>Sets the alignment of the specified cells.</td>
</tr>
<tr>
<td>GotoCell</td>
<td><code>Formn.Gridn.GotoCell c, r</code></td>
<td>The cell specified by column and row number will be in focus.</td>
</tr>
<tr>
<td>GotoColumn</td>
<td><code>Formn.Gridn.GotoColumn c</code></td>
<td>The column specified by column and row number will be in focus.</td>
</tr>
<tr>
<td>LockColumns</td>
<td><code>Formn.Gridn.LockColumns c</code></td>
<td>The specified number of columns remain visible when scrolling horizontally.</td>
</tr>
<tr>
<td>LockRows</td>
<td><code>Formn.Gridn.LockRows r</code></td>
<td>The specified number of rows remain visible when scrolling vertically.</td>
</tr>
<tr>
<td>SetCurrentColumn</td>
<td><code>Formn.Gridn.SetCurrentColumn c</code></td>
<td>Focus will be on the column indicated by c.</td>
</tr>
<tr>
<td>SetCurrentRow</td>
<td><code>Formn.Gridn.SetCurrentRow r</code></td>
<td>Focus will be on the row indicated by r.</td>
</tr>
<tr>
<td>SetGridDefault</td>
<td><code>Formn.Gridn.SetGridDefault</code></td>
<td>Enables you to specify the default values for a grid.</td>
</tr>
<tr>
<td>SetGridDefault</td>
<td><code>Formn.Gridn.SetGridDefault</code></td>
<td>Enables you to specify the default values for a grid.</td>
</tr>
</tbody>
</table>

**Syntax:**  
*How to Use the Mask Function With the ActiveX Grid*

To use the mask feature you have to call `UseMaskedEdit(1)` from the `OnEditStart` notification. Otherwise, the mask will have no effect.
Then use the following syntax string when coding in JavaScript:

```javascript
Formn.Gridn.QuickSetMask (0, 0, "#");
```

where:

```
#
```

Is any combination of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Digit (0 through 9). Entry is required. Plus (+) and minus (-) signs are not allowed.</td>
</tr>
<tr>
<td>9</td>
<td>Digit or space. Entry is optional. Plus (+) and minus (-) signs are not allowed.</td>
</tr>
<tr>
<td>L</td>
<td>Letter (A through Z). Entry is required.</td>
</tr>
<tr>
<td>?</td>
<td>Letter (A through Z). Entry is optional.</td>
</tr>
<tr>
<td>A</td>
<td>Letter or Digit. Entry is required.</td>
</tr>
<tr>
<td>a</td>
<td>Letter or Digit. Entry is optional.</td>
</tr>
<tr>
<td>&amp;</td>
<td>Any character or a space. Entry is required.</td>
</tr>
<tr>
<td>C</td>
<td>Any character or a space. Entry is optional.</td>
</tr>
<tr>
<td>. ,</td>
<td>Decimal placeholder and thousands, date, and time separators. The actual character used will depend on the regional settings specified in the Windows Control Panel.</td>
</tr>
<tr>
<td>: ;</td>
<td>Causes the character that follows to be displayed as a literal character. Used to display any of the characters listed in this table as literal characters. For example, \A is displayed as just A, &amp; is displayed as just &amp;, and so on.</td>
</tr>
</tbody>
</table>

For an example of how the Mask function can be used, see Setting a Mask in a Grid Cell on page 210.
**Example:** Displaying a New Font in a Grid Cell

To change the font in which the text in a cell appears, use the following VBScript within the push button event handler:

```vbnet
tfontid = Form1.Grid1.AddFont("Garamond", 14, 1)
Form1.Grid1.QuickSetFont 0,0,tfontid
Form1.Grid1.QuickSetText 0,0,"New Font"
Form1.Grid1.RedrawAll
```

When the push button is clicked, the first cell will display the text *New Font* in the Garamond font.

**Example:** Setting a Grid Cell to Read-Only Access

To set a grid cell to allow read-only access, use the following VBScript within the push button event handler:

```vbnet
Form1.Grid1.GetCell 0,0
Form1.Grid1.CellSetReadOnly True
Form1.Grid1.SetCell 0,0
Form1.Grid1.GetCell 0,0
```

When the push button is clicked, the first cell will be restricted to read-only access.

**Example:** Displaying a Message When a Grid Column Is Clicked

To display a message when a grid column is clicked, use the following VBScript within the OnLClicked grid event:

```vbnet
column = Form1.Grid1.GetCurrentColumn
message = "The column which was clicked is " & column
MsgBox message
```

When the grid is left-clicked, the message, *The column which was clicked is 0*, is displayed when column 1 is clicked. Column 1 will display 0, column 2 will display 1, and so on.

**Example:** Displaying a Message When a Grid Row Is Clicked

To display a message when a grid row is clicked, code the following VBScript within the OnLClicked grid event:

```vbnet
row = Form1.Grid1.GetCurrentRow
message = "The row which was clicked is " & row
MsgBox message
```

When the grid is left-clicked, the message, *The row which was clicked is 0*, is displayed when row 1 is clicked on. Row 1 will display 0, row 2 will display 1, and so on.
**Example: Setting the GoToCell Function**

To set the GoToCell function, code the following VBScript within the push button event handler:

```vb
Form1.Grid1.GoToCell 0, 0
Form1.Grid1.RedrawAll
```

When the push button is clicked, the first cell will be in focus.

**Example: Setting the GoToColumn Function**

To set the GoToColumn function, code the following VBScript within the push button event handler:

```vb
Form1.Grid1.GoToColumn 2
Form1.Grid1.RedrawAll
```

When the push button is clicked, the third column will be in focus.

**Example: Setting the SetCurrentColumn Function**

To set the SetCurrentColumn function, code the following VBScript within the push button event handler:

```vb
Form1.Grid1.SetCurrentColumn 3
Form1.Grid1.RedrawAll
```

When the push button is clicked, focus will be on the third column.

**Example: Setting the SetCurrentRow Function**

To set the SetCurrentRow function, code the following VBScript within the push button event handler:

```vb
Form1.Grid1.SetCurrentRow 3
Form1.Grid1.RedrawAll
```

When the push button is clicked, focus will be on the third row.

**Example: Setting a Mask in a Grid Cell**

To set an information mask in a cell, use the following VBScript:

```vb
Form1.Grid1.QuickSetMask (0, 1, "(###) ###-####");
```

The example allows the user to enter a ten digit number. The first three numbers appear in parentheses, and a dash appears after the next three numbers, as in a telephone number.
**Example: Setting Text in a Grid Cell**

To set the text of a cell, use the following VBScript within the push button event handler:

```
Form2.Grid1.GetCell 1,1
Form2.Grid1.SetCell 1,1
Form2.Grid1.RedrawAll
```

When the push button is clicked, the text of the selected cell will change to "NewText."

**Example: Setting Text Color in a Grid Cell**

To set the background color of a cell, use the following VBScript within the push button event handler:

```
Form1.Grid1.QuickSetTextcolor 2, 1, RGB(255,0,0)
Form1.Grid1.RedrawAll
```

This example specifies that the second cell in the first row will display the text color red.

**Example: Dynamically Changing Text Color in a Grid Cell**

To set the color of the text in a cell, use the following JavaScript with the OnSHLClicked event:

```
row=Form1.Grid1.GetCurrentRow();
numRows = Form1.Grid1.GetNumberRows();
numCols = Form1.Grid1.GetNumberColumns();
for(col=0; col < numCols; col++)
{
    Form1.Grid1.GetCell(col, row);
    Form1.Grid1.CellSetTextColor(0x0000ff);
    Form1.Grid1.SetCell(col, row);
    Form1.Grid1.Redrawall();
}
```

When the side heading is left-clicked, the text color in the selected cell will change.

**Note:** This JavaScript is recommended for use with the following events: OnSHDClicked, OnSHLClicked, OnSHRClicked, OnTHDClicked, OnTHLClicked, OnTHRClicked.
**Example:**  **Dynamically Changing Read-Only Access**

To dynamically set a cell to read-only access, use the following JavaScript with the OnTHLClicked event.

```javascript
numRows = Form1.Grid1.GetNumberRows();
for (row=0; row < numRows; row++)
{
    Form1.Grid1.GetCell(col, row);
    Form1.Grid1.CellSetReadOnly(1);
    Form1.Grid1.SetCell(col, row);
}
```

When the top heading is left-clicked, the selected cell becomes read-only.

**Note:** This JavaScript is recommended for use with the following events: OnSHDClicked, OnSHLClicked, OnSHRClicked, OnTHDClicked, OnTHLClicked, OnTHRClicked.

---

**Reference:**  **Property Functions in the Read/Write JS Grid**

<table>
<thead>
<tr>
<th>Name of Function</th>
<th>JavaScript</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Formn.Gridn.CellSetBackColor(col, row, color)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>where color is ,&quot;rgb(r,g,b)&quot;;</td>
<td></td>
</tr>
<tr>
<td>CellGetReadOnly</td>
<td>Formn.Gridn.CellGetReadOnly(col, row)</td>
<td>Retrieves the read-only setting, or sets the specified cell, row, or</td>
</tr>
<tr>
<td></td>
<td>Formn.Gridn.CellSetReadOnly(col, row, True</td>
<td>False)</td>
</tr>
<tr>
<td></td>
<td>Formn.Gridn.CellSetText(col, row, text)</td>
<td></td>
</tr>
<tr>
<td>Name of Function</td>
<td>JavaScript</td>
<td>Result</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CellSetFontStyle</td>
<td><code>Formn.Gridn.CellSetFontStyle(col, row, style)</code></td>
<td>Sets the font style for the specified cell. Font, font size and style can be set at one time using <code>CellSetFontStyle</code>.</td>
</tr>
<tr>
<td>CellGetTextColor</td>
<td><code>Formn.Gridn.CellGetTextColor(col, row)</code></td>
<td>Retrieves or sets the color of the text for the specified cell.</td>
</tr>
<tr>
<td></td>
<td><code>where color is ,&quot;rgb(r,g,b)&quot;</code></td>
<td></td>
</tr>
<tr>
<td>ColumnSetContentType</td>
<td><code>Formn.Gridn.ColumnSetContentType(col, contentType)</code></td>
<td>Allows HTML and images as well as text to be used as the content of the specified column. Usage notes:</td>
</tr>
<tr>
<td></td>
<td><code>Where contentType is:</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>❑ 0 - Text</td>
<td>This function should be set before the form loads using JSGridOnLoad, rather than in a control event.</td>
</tr>
<tr>
<td></td>
<td>❑ 1 - HTML</td>
<td>Use the GetHTMLfield command to retrieve the HTML values.</td>
</tr>
<tr>
<td></td>
<td>❑ 2 - Image</td>
<td>It is recommended that you set HTML columns as read-only to avoid having the HTML viewable when it is double-clicked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For images to display, the data in the cell must have a path to the file. For example, <code>/approot/ibi samp/sample.gif</code></td>
</tr>
<tr>
<td>GetCurrentColumn()</td>
<td><code>Formn.Gridn.GetCurrentColumn()</code></td>
<td>Identifies the column that is clicked on or selected by the user.</td>
</tr>
<tr>
<td>Name of Function</td>
<td>JavaScript</td>
<td>Result</td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
<td>--------</td>
</tr>
<tr>
<td>GetCurrentRow()</td>
<td><code>Formn.Gridn.GetCurrentRow()</code></td>
<td>Identifies the row that is clicked on or selected by the user.</td>
</tr>
<tr>
<td>GetNumberColumns</td>
<td><code>Formn.Gridn.GetNumberColumns()</code></td>
<td>Gets the total number of columns in the grid stack.</td>
</tr>
<tr>
<td>GetNumberRows</td>
<td><code>Formn.Gridn.GetNumberRows()</code></td>
<td>Gets the total number of rows in the grid stack.</td>
</tr>
<tr>
<td>OneClickEdit</td>
<td>`Formn.Gridn.OneClickEdit(True</td>
<td>False)`</td>
</tr>
<tr>
<td>SetCellFocus</td>
<td><code>Formn.Gridn.SetCellFocus(col, row)</code></td>
<td>Place the cursor on a Grid Cell.</td>
</tr>
</tbody>
</table>

**Note:** When specifying the number of columns, the row number is counted. When specifying current row and column, the first cell is 0,0.

To change the appearance of the JS Grid before the form is displayed, create your own JavaScript function called JSGridOnload, as shown in the following codeblock, and embed it onto the form.

```javascript
function JSGridOnload()
{
  Form1.Grid1.ColumnSetContentType(2,1);
}
```

All JS Grid functions, including ColumnSetContentType, CellSetBackColor and CellSetText, can be set using this method. Note that JavaScript is case-sensitive.

For more information on embedding JavaScripts into your Maintain form, refer to *Using Script Functions in Your Project* on page 133.

**Example:**  **Displaying New Fonts in a Grid Cell**

To change the font in which the text in a cell appears, use the following JavaScript within the push button event handler:

```javascript
Form1.Grid1.CellSetFont(2,1, "HERALD");
```

When the push button is clicked, the cell in column 2 row 1 will display the text in the Herald font.
Example: Displaying New Font Styles in a Grid Cell
To change the font style in which the text in a cell appears, use the following JavaScript within the push button event handler:

```
font-style: italic; text-decoration: underline;"");
```

When the push button is clicked, the cell in column 1 row 1 will display the text in italics with a size of 12 and underlined.

CellSetFontStyle can also be used to set only the size of the font.

Example: Setting the Focus on a Cell
To set the focus on a cell, code the following JavaScript within the push button event handler:

```
Form1.Grid1.SetCellFocus(0,0);
```

When the push button is clicked, the first cell will be in focus.

Example: Getting the Current Column and Current Row
To get the current column or row, code the following JavaScript within the grid event handler for a grid event such as OnRClicked:

```
var crow = Form1.Grid1.GetCurrentRow();
var ccol = Form1.Grid1.GetCurrentColumn();
alert("You are on row " + crow + " and column " + ccol);
```

When right-clicking on a particular cell, the alert will display the column and row numbers for that cell.

Example: Getting the Total Number of Rows and Columns
To get the total number of rows and/or columns in the grid, use the following JavaScript code from within the grid event handler for a grid event such as OnRClicked:

```
var nrow = Form1.Grid1.GetNumberRows();
var ncol = Form1.Grid1.GetNumberColumns();
alert("This grid has " + " " + nrow + " rows and " + ncol + " columns");
```

When right-clicking on the grid, the alert will display the total number of rows and columns in the grid.
**Example: Setting Text in a Grid Cell**

To set the text of a cell, use the following JavaScript within the push button event handler:

```javascript
var x = Form1.Grid1.CellGetText(1,1);
alert(x);
```

When the push button is clicked, the alert will show the starting text of the selected cell and the text will change to *NewText*.

**Example: Setting Text Color in a Grid Cell**

To set the text color of a cell, use the following JavaScript within the push button event handler:

```javascript
Form1.Grid1.CellSetTextColor(0,0,"rgb(255,0,0)");
```

This example specifies that the first cell will display the text color in red.

**Example: Setting Back Color in a Grid Cell**

To set the background color of a cell, use the following JavaScript within the push button event handler:

```javascript
Form1.Grid1.CellSetBackColor(0,0,"rgb(255,0,0)");
```

This example specifies that the first cell will display the background color in red.

**Example: Setting a Grid Cell to Read-Only Access**

To set a grid cell to allow read-only access, use the following JavaScript within the push button event handler:

```javascript
Form1.Grid1.CellSetReadOnly(0,0, "TRUE");
```

When the push button is clicked, the first cell will be restricted to read-only access.

**Example: Setting Content Type for a Grid Column**

To set the content type for a column, create a JSGridOnLoad function that uses the following JavaScript to specify the content prior to the display of the form:

```javascript
Form1.Grid1.ColumnSetContentType(1,2);
```

This example sets the content type of the second column to Image. The data for the column is a variable that must be defined with a path to the image file, as in the following:

```javascript
Compute stk(i).pic/a100="/approot/ibisamp/logo.gif";
```
Using Group Boxes

A group box places a border on the form. You can place this border around a group of related controls to help orient your end users. Group boxes are usually cosmetic, but can help convey logical groups of controls.

**Procedure: How to Add a Group Box Control to Your Form**

1. Select the *Group box* control in the Controls palette.
2. Draw a rectangle on your form where you want your group box at approximately the size you want.
3. Type the text you want your group box to have and press Enter. The text should be selected automatically when you created the group box.
4. If necessary, readjust the size and placement of your group box.

If you will be adjusting the placement of your group box and the controls it surrounds, consider grouping these objects together. For more information, see *Grouping Controls* on page 101.

Changing Group Box Properties

When you select your group box, you will see a list of group box properties in the property sheet. Changing these properties will change what your group box looks like and what it does at run time.

Many of the styling properties can also be changed using a cascading style sheet. For more information on Cascading Style Sheets, see *Using Cascading Style Sheets* on page 113.

Choose your property based on the task you wish to perform:

**If you want to change the label at the top of the group box,** use the Text Property. For more information, see the *Text Property* on page 292.

**If you want to change the name of the group box that identifies it to the procedure,** use the *(Name) Property*. For more information, see the *(Name) Property* on page 267.

**If you want to change the size or location of the group box,** use the Bottom Property, Left Property, Right Property, Top Property, or use a style sheet. For more information, see the *Bottom Property* on page 273, *Left Property* on page 286, *Right Property* on page 289, and *Top Property* on page 293. You can also move or resize the group box directly in the form.

**If you want to change the color of the group box,** use the *BorderColor Property* on page 272 to determine the color of the border. Use the *ForeColor Property* on page 279 to determine the color of the text in the label. You can also use a style sheet.
If you want to change the label font, use a style sheet or use the Font Property. For more information, see the Font Property on page 279.

If you want to change the alignment of the label to left-justified, centered, or right-justified, use the Alignment Property or use a style sheet. For more information, see Alignment Property on page 267.

If you want to change the border type or width, use the Border Property to determine the type of border (none, normal 3D, and so on), use the BorderWidth Property to determine the width of the border, or use a style sheet. For more information, see the Border Property on page 271 and the BorderWidth Property on page 273.

If you want to change what the cursor looks like when it is on top of the group box, use the CursorPointer Property or use a style sheet. For more information, see the CursorPointer Property on page 276.

If you want to make the group box inactive or make it invisible, use the Enabled Property to determine whether the group box is active or not. If the group box is inactive, it will be dimmed out and nothing will happen when the end user clicks it. You can also use the Visible Property to determine whether the group box is visible to the end user or use a style sheet. For more information, see the Enabled Property on page 278 and the Visible Property on page 294.

If you want to display a tooltip when the cursor is on the top of the group box, use the ToolTipText Property. The tooltip applies to the whole inner area, including any controls within it, unless these controls have their own tooltips. For more information, see the ToolTipText Property on page 293.

If you want to assign a Help topic to the group box, use the Help Property. For more information, see Assigning Help to Your Forms and Controls on page 263.

If you want to move the group box to another layer, use the Layer Property. For more information, see Layering Controls on page 103.

If you want to control the display order of the group box when more than one control is overlaid, use the ZIndex Property. For more information, see the ZIndex Property on page 294.

Using HTML Objects

HTML Object controls enable you to provide your own HTML code and place it in any location on the form. You can use HTML tags, including Java applets, JavaScript, VBScript, and others. You can also code an Exec to a WebFOCUS report, or a table services report and integrate its output into a form.

The position of the top-left corner of the HTML object on the form determines the starting position of the HTML code after it appears in a browser.
One of the most common uses for an HTML Object is to display the formatted results of a WebFOCUS report. For an example of this technique, see the Maintain Getting Started manual.

Another use for an HTML Object is to display the resulting page of a web link in a frame. For more information on this technique, see Defining Events and Event Handlers on page 125.

**Procedure: How to Place an HTML Object on Your Form**

1. Select the HTML Object control in the Controls palette.
2. Draw a rectangle on your form where you want your HTML Object at approximately the size you want.
3. Giving your HTML Object a more meaningful name than HTMLn is optional, but recommended.
4. If necessary, readjust the size and placement of your HTML Object.
5. Double-click the HTML Object or select the Content property to open the HTML Content Source dialog box. You will use the dialog box to determine the data source for your HTML Object.
6. To directly set the content of the HTML Object, select As entered below, and then enter the HTML code in the box.

To assign the content of the HTML Object to a variable, select From a variable, then select a variable or data source stack column. You can create a new variable or data source stack by clicking New variable or New data source stack. For more information, see How to Create a Variable in a Procedure on page 68 or How to Create a Data Source Stack Explicitly Using the Stack Editor on page 73.

7. Click OK.

If you assign the content of the HTML Object to a variable or data source stack column, you will see the name of the variable or data source stack column in the HTML Object.
**Reference:** HTML Content Source Dialog Box

Use the HTML Content Source dialog box, shown in the following image, to define the contents of the HTML Object.

This dialog box contains the following elements:

**Insert HTML content**

Select *As entered here* to set the contents directly.
Select *From a variable* to set the contents from a variable, as shown in the following image.

**New variable**

This is available only if you select *From a variable*. This option opens the New Variable dialog box, where you can create a variable with the data type Stack of.

**New data source stack**

This is available only if you select *From a variable*. This option opens the Stack Editor, where you can create a stack.

**List of data source stacks and variables in your procedure**

This is available only if you select *From a variable*. This option contains a list of the existing stacks and variables in your procedure.

Either select a variable, or expand a data source stack and select a column.
Changing HTML Object Properties

When you select your HTML Object, you will see a list of HTML Object properties in the property sheet. Changing these properties will change what your HTML Object looks like and how it behaves at run time.

Many of the styling properties can also be changed using a Cascading Style Sheet. For more information on Cascading Style Sheets, see Using Cascading Style Sheets on page 113.

Choose your property based on the task you wish to perform:

If you want to change the contents of the HTML Object, use the Content Property to open the HTML Content Source Dialog Box on page 220. You can also change the contents of the stack column or variable it is bound to.

If you want to change the name of the HTML Object that identifies it to the procedure, use the (Name) Property. For more information, see the (Name) Property on page 267.

If you want to change the size or location of the HTML Object, use the Bottom Property, Left Property, Right Property, Top Property, or use a style sheet. For more information, see the Bottom Property on page 273, Left Property on page 286, Right Property on page 289, and Top Property on page 293. You can also move or resize the HTML Object directly in the form.

Note: If the Overflow property is set to Visible, the HTML Object will be as wide and as long as necessary in order to display everything.

If you want to change the color of the HTML Object, use the BackColor Property on page 269 to determine the background color, the ForeColor Property on page 279, or use a style sheet.

If you want to change the text font, use the Font Property on page 279 or use a style sheet.

If you want to add a border to the HTML Object, use a style sheet or the Border Property on page 271 to determine whether you have a border and what type of border it is. The BorderColor Property on page 272 allows you to determine the color of the border. The BorderWidth Property on page 273 allows you to determine the width of the border. You may need to increase the BorderWidth and change BorderColor in order to take full advantage of some of the border types.

If you want to make the HTML Object inactive or make it invisible, use the Enabled Property to determine whether the HTML Object is active or not. If the HTML Object is inactive, it will be dimmed out and nothing will happen when the end user clicks it. You can also use the Visible Property to determine whether the HTML Object is visible to the end user or use a style sheet. For more information, see the Enabled Property on page 278 and the Visible Property on page 294.

If you want to change what the cursor looks like when it is on top of the HTML Object, use the CursorPointer Property on page 276 or use a style sheet.
If you want to display a tooltip when the cursor is on the top of the HTML Object, use the ToolTipText Property on page 293.

If you want to assign a Help topic to the HTML Object, use the Help Property. For more information, see Assigning Help to Your Forms and Controls on page 263.

If you want to move the HTML Object to another layer, use the Layer Property. For more information, see Layering Controls on page 103.

If you want to control the display order of the HTML Object when more than one control is overlaid, use the ZIndex Property. For more information, see the ZIndex Property on page 294.

Using Images

WebFOCUS Maintain makes it easy for you to add .jpg or .gif images to your forms. You can add images in one of two ways:

- As a background image on your form (using the BackgroundImage property for your form). Background images are for display only and you can take advantage of WebFOCUS Maintain tiling feature.

- Using the Image control. You would insert an image as a control if you wanted your end users to be able to click it (although image controls can be purely decorative also).

Images are included in your project as resources. Resources are part of your project, but you do not edit them directly using the Maintain Development Environment. However, WebFOCUS Maintain will take care of deploying them to the correct location so that your application will run properly.

Procedure:  How to Add an Image to Your Form

1. If you want to insert the image using the Image control, select the Image control in the Controls palette and select an area on the form approximately where you want the image to go.

   or

   If you want to insert the image as a background image, display the properties for the form in the property sheet and double-click the BackgroundImage property.

2. In the Image Source dialog box, determine whether you want to directly enter the name of your image (From the resource selected here), or use a variable (From a variable).

3. If you choose to select an image resource here, select an image from the list of available images in your project.
If the image you want is not part of your project, click New to use the Resource Wizard. See *How to Add an Image to Your Project as a Resource* on page 224 for more information.

or

If you choose to use a variable, select a variable or data source stack column from the list of available variables. The variable or data source stack column must contain a path to an image file. If the path contains backslashes, use double backslashes (for example, C:\ibi_img\graphic.gif).

If you want to create a variable, click New variable or New data source stack. For more information, see *How to Create a Variable in a Procedure* on page 68 and *How to Create a Data Source Stack Explicitly Using the Stack Editor* on page 73.

4. For background images only, select a tiling style. When you tile your image, WebFOCUS Maintain places multiple copies of it next to each other on the form.

5. Click OK.

6. Images only. Optional, but recommended. Giving your image a more meaningful name than Image is optional, but recommended.

7. (Images only.) If necessary, readjust the size and placement of your image.

8. If you used a variable to insert the image, open the variable definition in the Procedure Editor, and enter the path to the image file.

**Procedure: How to Add an Image to Your Project as a Resource**

1. In the Image Source dialog box, select the From the resource selected here radio button and click New.

   WebFOCUS Maintain opens the Resource Wizard, a series of windows that guide you through adding an image to your project.

2. Enter the path to the image. If you want to see what the image looks like, click Preview.

   If you prefer to browse for the image, click the ellipsis button. The Resource Wizard will automatically preview the image you select.

   Then click Next.

3. Specify a unique name for your image. Then click Finish.

4. You may see a dialog box informing you that WebFOCUS Maintain needs to make a copy of the image in the project directory. Click OK.
Reference: Image Source Dialog Box

Use the Image Source dialog box, shown in the following image, to insert images into your forms, either using the Image control, or as the BackgroundImage property for your form.

Insert image

Select *From the resource selected here* to select an image directly. When you select this option, you will see a list of available images and a *New* button that enables you to add new images.

Select *From a variable* to set an image to be the value of a variable. This enables you to determine at run time what image should appear. When you select this option, you will see a list of available variables in your procedure.

List of available images

If you select *From the resource selected here*, you will see this list of available images. Initially, `<None>` is selected. Select the image you want to display. Click *New* to add images to the list.
New

Opens the Resource Wizard so that you can add new images to your project.

Tiling style

(For background images on forms only.) Determines whether repeated copies of the image appear on the form. If you wish to use a background image that repeats (for example, a pattern of dots), you can make your form download and run faster by using multiple copies of a smaller image.

You have four choices for tiling:

- 0 - Full places multiple copies of the image across the entire area of the form.
- 1 - Vertical repeats the image from the top edge of the form down the entire length.
- 2 - Horizontal repeats the image from the left edge of the form across the entire width.
- 4 - None places one copy of the image at the upper-left corner of the form window.

New variable

This is available only if you select From a variable. This option opens the New Variable dialog box, where you can create a variable.

New data source stack

This is available only if you select From a variable. This option opens the Stack Editor, where you can create a stack.

List of data source stacks and variables in your procedure

This is available only if you select From a variable. This option contains a list of the existing stacks and variables in your procedure, as shown in the following image.
Either select a variable, or expand a data source stack and select a column.

**Data sources - Current Area**

This is available only if you select *From a variable*. This option lists the fields from the data sources used in this procedure.

**Note:** We recommend that you do not use the Current Area. Data source stacks are a superior way of accessing and manipulating data source values, and they function more intuitively than the Current Area.

**Tips for Using Images in Your Applications**

- Images are very easy to find on the web. There are a lot of websites out there that provide free images for buttons and backgrounds.
Using Images

- However, do not overuse images, particularly on the web. Know the bandwidth at your site and keep in mind that any image that you put on your form will have to be downloaded by the browser at run time. If this is an Internet application, you do not know how fast the modems of your end users will be.

  Browsers download HTML/DHTML first and then images. In Internet Explorer, the status bar will say "N files remaining to download." This is because the browser has the HTML already, but it is retrieving the images. Instead of downloading several images, consider using one larger composite image and defining an image map. For more information, see Changing Image Properties on page 228.

- Use smaller image sizes and the appropriate image type (.jpgs are better for photos, .gifs for drawings).

- For background images, consider using a smaller image and setting a tiling style (where multiple copies of the image are placed next to each other on the form). This reduces the amount of time it takes to download images.

**Changing Image Properties**

When you select your image, you will see a list of image properties in the property sheet. Changing these properties will change what your image looks like and what it does at run time.

Many of the styling properties can also be changed using a Cascading Style Sheet. For more information on Cascading Style Sheets, see Using Cascading Style Sheets on page 113.

Choose your property based on the task you wish to perform:

**If you want to change the name of the image that identifies it to the procedure**, use the (Name) Property. For more information, see the (Name) Property on page 267.

**If you want to change which image will appear**, use the Image Property to open the Image Source Dialog Box on page 225.

**If you want to change the size of the location or size of the image**, use a style sheet or resize the image directly in the form. You can also use the Stretched Property on page 291 and the Bottom Property on page 273, Left Property on page 286, Right Property on page 289, and Top Property on page 293.

**If you want to add a border to the image**, use a style sheet or the Border Property on page 271 to determine whether you have a border and what type of border it is. The BorderColor Property on page 272 allows you to determine the color of the border. The BorderWidth Property on page 273 allows you to determine the width of the border.
If you want to change what the image looks like when the end user selects the image, use the ImageDown Property. For more information, see Changing Images at Run Time Using ImageDown and ImageOver on page 232.

If you want to make different regions of the image clickable, use the Map Property. For more information, see Using Image Maps on page 229.

If you want to make the image inactive or make it invisible, use the Enabled Property to determine whether the image is active or not. If the image is inactive, nothing will happen when the end user clicks it. You can also use the Visible Property to determine whether the image is visible to the end user or use a style sheet. For more information, see the Enabled Property on page 278 and the Visible Property on page 294.

If you want to change what the cursor looks like when it is on top of the image, use the CursorPointer Property on page 276 or use a style sheet.

If you want to change what the image looks like when the cursor is on top of the image, use the ImageOver property. For more information, see Changing Images at Run Time Using ImageDown and ImageOver on page 232.

If you want to display a tooltip when the cursor is on top of the image, use the Alt Property on page 268 or the ToolTipText Property on page 293.

If you want to display a tooltip when the cursor is on the top of the image, use the ToolTipText Property on page 293.

If you want to assign a Help topic to the image, use the Help Property. For more information, see Assigning Help to Your Forms and Controls on page 263.

If you want to move the image to another layer, use the Layer Property. For more information, see Layering Controls on page 103.

If you want to control the display order of the image when more than one control is overlaid, use the ZIndex Property. For more information, see the ZIndex Property on page 294.

Using Image Maps

Using the Map property, you can define separate clickable areas on an image. At run time, end users can click on separate areas of the image to trigger different actions. This enables you to use one image instead of several, thus making your application smaller and faster to download.

After you define an image map, when you open the Event Handler editor for the image, you will see new events that correspond to each of the areas in the image map.
**Procedure: How to Define an Image Map**

1. Select the image control you want to apply an image map to and select the Map property. WebFOCUS Maintain opens the Image Map dialog box.

2. Click New to create an image map.

3. Define clickable areas in the image using one of the three tools. You can define areas shaped like a circle, rectangle, or polygon.

   ![Circle, Rectangle, Polygon Icons]

   To delete an area, use the Delete and Delete all buttons.

4. If you wish, rename your areas and apply tooltips to each.

5. Click OK.

6. Open the Event Handler editor for the image.

7. In the list of events for the image, you will see Click events for each of the areas you defined in the Image Map dialog box.

8. Define event handlers for each of these events.
**Reference: Image Map Dialog Box**

Use the Image Map dialog box, shown in the following image, to define clickable locations in an image.

This dialog box includes the following options:

**Available maps**

Contains a list of the maps available for this image resource (you can have more than one). To use one, select it. To rename one, select it and type a new name.

**New**

Creates a new image map.

**Image map area**

Create and edits your image areas that define what end users can click.
Enables you to select areas for editing.

Enables you to define circle-shaped areas on the image.

Enables you to define rectangle-shaped areas on the image.

Enables you to define irregular, polygon-shaped areas on the image.

Deletes a selected area.

Deletes all areas defined in this image map.

**Description of selected area**

Describes what shape and size the selected area has.

**Area name**

Is the name of the selected area. To rename an area, select it and type a new name here.

**Tooltip**

Enables you to enter a tooltip for the selected area.

**Changing Images at Run Time Using ImageDown and ImageOver**

If you are defining your image to be clickable that is, an end user clicking the image causes some action to occur, then you may wish to provide visual feedback to your end users that the image has been clicked. You may also want to have the image change when end users move their cursors over the image, indicating that the image is clickable. You can do this using the ImageDown and ImageOver properties.

To use another image as an ImageDown or ImageOver, we recommend the following:

- The image should be the same size as the original image.
The image should be similar to the original image. For example, ImageDown images are frequently negative versions of the original image.

**Procedure: How to Change an Image When End Users Click It**

1. Select the image.
2. Double-click the ImageDown or ImageOver property to open the Image Source dialog box.
   or
   Right-click the image and in the shortcut menu, click Change down image or Change over image.
3. Follow steps 2 through 6, as shown in *How to Add an Image to Your Form* on page 223.

**Using Java Applets**

WebFOCUS Maintain makes it easy for you to add Java applets to your forms. Java applets are included in your project as resources. Resources are part of your project, but you do not edit them directly using the Maintain Development Environment. However, WebFOCUS Maintain will take care of deploying them to the correct location so that your application will run properly.

When you define a Java applet as a resource to your project, you can define some or all of its parameters and default values for these parameters. Often, Java applets are accompanied by a text file documenting the names and default values of these parameters.

When you place a Java applet control on your form, you can assign new values to these parameters. You can also dynamically define new parameters or change the values of existing parameters during run time.

**Note:** If your Java applet involves using other Java classes, you must either package the applet with these classes, or inform your end users to include the location of these classes in their CLASSPATH. See the documentation for Internet Explorer, for more information.

**Procedure: How to Place a Java Applet on Your Form**

1. Select the Java applet control in the Controls palette.
2. Draw a rectangle on your form to define the approximate size and location of your Java applet.
   The Parameters dialog box opens.
3. Select a Java applet from the list of available Java applets in your project.
4. If necessary, specify new parameter values for the Java applet. You can use the parameters and default values you specified when you added the resource to your project, or you can override these parameters. If a control provides no value for a parameter, the parameter is ignored.

5. Click OK.

6. If necessary, readjust the size and placement of your Java applet.

7. Giving your Java applet a more meaningful name than JavaAppletn is optional, but recommended.

**Procedure: How to Add a Java Applet to Your Project as a Resource**

1. In the Parameters dialog box, click New.

   WebFOCUS Maintain opens the Resource Wizard, a series of windows that guide you through adding a Java Applet to your project.

2. Enter the path to the Java applet. If you prefer to browse for the Java applet, click the ellipsis button ...

   **Note:** Be sure to use the correct case when specifying your Java applet name. As a rule, the .class extension should be lowercase.

   Optionally, add a description for the Java applet.

   Then click Next.

3. Specify parameters and, optionally, default values for the Java applet. You do not have to define all of the Java applet parameters, just the ones that you want to supply values for in your application.

   Then click Next.

4. Specify a unique name for your Java applet.

   Then click Finish.
5. You may see a dialog box informing you that WebFOCUS Maintain needs to make a copy of the Java applet in the project directory. Click OK.

**Syntax:**  
*How to Change Java Applet Parameters Dynamically at Run Time*

To change the name of a Java applet parameter or add a new parameter, use the following syntax:

```java
Form.JavaApplet.params(n).Name="name"
```

To change the value for a Java applet parameter, use the following syntax

```java
Form.JavaApplet.params(n).Value="value"
```

where:

- **Form**
  - Is the name of the form on which the Java applet control is residing.

- **JavaApplet**
  - Is the name of the Java applet control (not the Java applet).

- **n**
  - Is the position of the parameter in the list of parameters defined to WebFOCUS Maintain.
Reference: Parameters Dialog Box

Use the Java Applet Parameters dialog box, shown in the following image, to define the behavior of a Java applet control that you have placed on your form.

This dialog box includes the following options:

**Use the Java applet from this resource**

Contains a list of Java applet resources that are included in this project.

**New**

Opens the Resource Wizard, where you can migrate or link to a new Java applet.

**Parameters**

Contains a list of the parameters for this Java applet.

**Value**

Contains the default value for the parameters listed. You can override these values if you wish.

Enables you to edit the default value for the selected parameter.
Deletes a default value for the selected parameter.

Moves a selected parameter up in the list of parameters.

Moves a selected parameter down in the list of parameters.

**Reference:** Enter a Parameter Dialog Box

Use the Enter a Parameter dialog box, shown in the following image, to define parameters for Java Applets, and to assign values to parameters when you place Java applets on your forms.

Also use the Enter a Parameter dialog box to define amper variables and values when defining a WebFOCUS report as a web link. For more information, see *Defining Events and Event Handlers* on page 125.

![Enter a Parameter Dialog Box](image)

This dialog box has the following components:

**Parameter name**

If you are defining parameters for a Java applet, WebFOCUS report, enter its name here. Usually, Java applets come with text documentation that lists the names of the parameters.

If you are assigning a new value to a parameter for a Java applet control, you cannot edit this value.

**Default value/Value**

If you are defining parameters for a Java applet, enter the default value here.
If you are assigning a new value to a parameter for a Java applet control, enter the new value here.

**Changing Java Applet Properties**

When you select your Java applet, you will see a list of Java applet properties in the property sheet. Changing these properties will change what your Java applet looks like and what it does at run time.

Many of the styling properties can also be changed using a Cascading Style Sheet. For more information on Cascading Style Sheets, see *Using Cascading Style Sheets* on page 113.

Choose your property based on the task you wish to perform:

- **If you want to change the name of the Java applet that identifies it to the procedure**, use the (Name) Property. For more information, see the *(Name) Property* on page 267.

- **If you want to change the Java applet or if you want to change the values for Java applet parameters**, use the Applet Property to open the Parameters Dialog Box on page 236.

- **If you want to change the size of the location or size of the Java applet**, use a style sheet or resize the Java applet directly in the form. You can also use the Bottom Property on page 273, Left Property on page 286, Right Property on page 289, and Top Property on page 293.

- **If you want to add a border to the Java applet**, use a style sheet or the Border Property on page 271 to turn the Java applet borders on and off. The BorderColor Property on page 272 allows you to determine the color of the border. The BorderWidth Property on page 273 allows you to determine the width of the border.

- **If you want to make the image inactive or make it invisible**, use the Enabled Property to determine whether the Java applet is active or not. If the Java applet is inactive, nothing will happen when the end user clicks it. You can also use the Visible Property to determine whether the Java applet is visible to the end user or use a style sheet. For more information, see the Enabled Property on page 278 and the Visible Property on page 294.

- **If you want to change what the cursor looks like when it is on top of the Java applet**, use the CursorPointer Property on page 276 or use a style sheet.

- **If you want to display a tooltip when the cursor is on the top of the Java applet**, use the ToolTipText Property on page 293.

- **If you want to assign a Help topic to the Java applet**, use the Help Property. For more information, see *Assigning Help to Your Forms and Controls* on page 263.

- **If you want to move the Java applet to another layer**, use the Layer Property. For more information, see *Layering Controls* on page 103.
If you want to control the display order of the Java applet when more than one control is overlaid, use the ZIndex Property. For more information, see the ZIndex Property on page 294.

Using Lines

You can place vertical or horizontal lines on your form for visual effect, for example, to separate groups of controls.

Procedure: How to Place a Line on Your Form

1. Select the Line control in the Controls palette.
2. Click and hold down the mouse button at the beginning point for your line.
3. Move the cursor to the ending point for your line and release the mouse button.

   Note: You can only draw horizontal or vertical lines.

4. If necessary, readjust the size and placement of your line.

Changing Line Properties

When you select your line, you will see a list of line properties in the property sheet. Changing these properties will change what your line looks like and what it does at run time.

Many of the styling properties can also be changed using a Cascading Style Sheet. For more information on Cascading Style Sheets, see Using Cascading Style Sheets on page 113.

Choose your property based on the task you wish to perform:

If you want to change the name of the line that identifies it to the procedure, use the (Name) Property. For more information, see the (Name) Property on page 267.

If you want to change the size of the location or size of the line, use a style sheet or resize the line directly in the form. You can also use the Bottom Property on page 273, Left Property on page 286, Right Property on page 289, and Top Property on page 293.

If you want to change the color of the line or width, use a style sheet or the ForeColor Property on page 279 to determine the color of the line. Use the PenWidth Property on page 289 to determine the width of the line.
If you want to make the line inactive or make it invisible, use the Enabled Property to determine whether the line is active or not. If the line is inactive, nothing will happen when the end user clicks it. You can also use the Visible Property to determine whether the line is visible to the end user or use a style sheet. For more information, see the Enabled Property on page 278 and the Visible Property on page 294.

If you want to change what the cursor looks like when it is on top of the line, use the CursorPointer Property on page 276 or use a style sheet.

If you want to display a tooltip when the cursor is on the top of the line, use the ToolTipText Property on page 293.

If you want to assign a Help topic to the line, use the Help Property. For more information, see Assigning Help to Your Forms and Controls on page 263.

If you want to move the line to another layer, use the Layer Property. For more information, see Layering Controls on page 103.

Using Menus

The menu control enables you to create a menu bar with pull-down menus in your form. You may create only one menu bar per form.

Using the menu control requires two steps:

- Creating a menu bar that includes all the pull-down menus you wish to display. For example, menu bars in many applications include the File, Edit, Window, and Help menus. For more information, see How to Place a Menu Bar on Your Form on page 240.

- Creating the menu items and their associated commands for each pull-down menu in the Menu Items dialog box. For more information, see How to Create Pull-Down Menus and Submenus on page 241.

Procedure: How to Place a Menu Bar on Your Form

1. Click the Menu control in the Controls palette.
2. Draw a rectangle on your form approximately where you want your menu bar to be, at approximately the size you want. You can create a horizontally or vertically aligned menu.
3. Giving Menu Bar a more meaningful name than MenuBar is optional, but recommended.
4. If necessary, readjust the size and placement of your menu bar.
5. Double-click the menu bar rectangle or select Properties for the menu control to open the Menu Items dialog box. Specify all the pull-down menus and submenus you wish to include. For more information on specifying menu items and their associated commands, see How to Create Pull-Down Menus and Submenus on page 241.

6. When all desired menu items have been specified in the Menu Items dialog box, click OK.

Procedure: How to Create Pull-Down Menus and Submenus

To create a pull-down menu or submenu, complete the following steps from the Menu Items dialog box. This procedure assumes that you have already placed a menu bar on your form. For more information on how to place a menu bar on your form, see How to Place a Menu Bar on Your Form on page 240.

1. Double-click the menu bar you have placed on your form in order to access the Menu Items list box.

2. To create a pull-down menu, click the New button.

3. Type the name of the pull-down menu as you want it to display on the menu bar in the Text box.

4. Type the name you want to use to identify the menu bar to your procedure in the Name box.

5. To create a submenu, select the name of the pull-down menu to which you are adding the submenu. Repeat steps 1-3 for each submenu desired.

6. Repeat steps 1-4 until all desired pull-down menus and submenus have been specified in the Menu Items dialog box.

7. Click OK.

8. Assign a Click action to the lowest level menu items using the Event Handler editor. You can access the Event Handler editor one of the following ways:
   - Right-click the menu bar and select Edit event handlers from the pop-up menu.
   - Select the Events tab in the Properties box and click the Browse button.

For more information on assigning actions to events, see Defining Events and Event Handlers on page 125.

Example: Creating a Menu Bar With Pull-Down Menus and Submenus

The following example demonstrates how to create a pull-down menu that contains submenus.

1. Select the Menu control from the Controls Palette.
2. Draw a horizontal rectangle on your form in order to create a horizontal menu bar, as shown in the following image.

**Note:** You can change the orientation of the menu control by changing the Orientation property in the property sheet.

3. Double-click the menu bar to open the Menu Items dialog box, shown in the following image.
4. Create menu items on your form by selecting MenuBar1 and clicking the New button. Type the text that you want to appear on the menu bar in the Text box, and the name you will use to identify the menu item in the procedure in the Name box.

5. Create submenus by selecting a menu item and clicking the New button. Type the name you want to give your submenu (for example, Menu5) in the Text box, and type the name you will use to identify the menu item in the procedure in the Name box. When all submenus have been specified, click OK.
The highest level menu items will appear on your form, as shown in the following image. Readjust the size and placement of your menu bar, if necessary. An arrow on a menu item indicates that the menu item contains submenus.

6. For each lowest level menu item, assign a click action using the Event Handler editor.

For more information on assigning actions to events, see *Defining Events and Event Handlers* on page 125.

**Reference:** **The Menu Items Dialog Box**

Use the Menu Items dialog box, shown in the following image, to determine the menu items that will appear on your menu bar, including pull-down menus and submenus.
This dialog box includes the following options:

**Text**

Enables you to specify the prompt text for each item.

**Name**

Enables you to specify the name of each item.

**+**

Enables you to add a selected item to the list of menu items.

**-**

Deletes a selected item from the list of items.

**↑**

Moves a selected item up in the list of items.

**↓**

Moves a selected item down in the list of items.

**Changing Menu Bar Properties**

When you select your menu control, you will see a list of control properties in the property sheet. Changing these properties will change what your menu bar looks like and what it does at runtime.

Many of the styling properties can also be changed using a Cascading Style Sheet. For more information on Cascading Style Sheets, see *Using Cascading Style Sheets* on page 113.

Choose your property based on the task you wish to perform:

**If you want to change the name of the menu bar that identifies it to the procedure,** use the (Name) Property. For more information, see the *(Name) Property* on page 267.

**If you want to change the color of the menu bar,** use a style sheet or the ForeColor Property on page 279 to determine the color of the text in the box. Use the BackColor Property on page 269 to determine the color of the box.

**If you want to add a border to the menu bar,** use a style sheet or the Border Property on page 271.
**If you want to add a border to an individual menu item**, use a style sheet or the *ItemBorder Property* on page 285.

**If you want to change the size of the location or size of the menu bar**, use a style sheet or resize the line directly in the form. You can also use the *Bottom Property* on page 273, *Left Property* on page 286, *Right Property* on page 289, and *Top Property* on page 293.

**If you want to change what the cursor looks like when it is on top of the menu bar**, use the *CursorPointer Property* on page 276 or use a style sheet.

**If you want to change the color of the text or the background when the cursor is on top of the menu bar**, use the *BackColorOver Property* on page 270 and the *ForeColorOver Property* on page 280. You can also use a style sheet.

**If you want to make the menu bar inactive or make it invisible**, use the Enabled Property to determine whether the is active or not. If the line is inactive, nothing will happen when the end user clicks it. You can also use the Visible Property to determine whether the line is visible to the end user or use a style sheet. For more information, see the *Enabled Property* on page 278 and the *Visible Property* on page 294.

**If you want to assign a Help topic to the menu control**, use the Help Property. For more information, see *Assigning Help to Your Forms and Controls* on page 263.

**If you want to move the menu bar to another layer**, use the Layer Property. For more information, see *Layering Controls* on page 103.

**If you want to change the alignment of your menu bar from vertical to horizontal, or vice versa**, use the *Orientation Property* on page 287.

**If you want to display a tooltip when the cursor is on the top of the menu controls**, use the *ToolTipText Property* on page 293.

**Using Radio Buttons**

Radio buttons enable end users to make a single choice among a group of options. They are best used for a static group of options that change rarely or not at all (for example, gender, or day of the week).

The contents of the list are determined dynamically at run time by one of the following:

- A list you supply at development time (entered manually in the List Items dialog box).
- The contents of a data source stack or variable.
- The list of valid values defined for a data source field in a data source description.
You define the radio button contents using the List Source dialog box, described in *List Source Dialog Box* on page 176. You define how to store the value returned from the radio buttons using the Binding the Selection Result dialog box, described in *Binding the Selection Result Dialog Box* on page 179.

**Procedure: How to Place a Radio Button on Your Form**

1. Select the *Radio button* control in the Controls palette.
2. Draw a rectangle on your form approximately where you want your radio buttons to be at approximately the size you want.
3. Giving you radio buttons a more meaningful name than RadioButton
   is optional, but recommended.
4. If necessary, readjust the size and placement of your radio buttons.
5. Double-click the radio buttons or select the *ListItems* property to open the List Source dialog box. Use this dialog box to determine the items in your list.
6. You have four options for defining the items in your radio buttons:
   - **Manually.** Select *As entered here* and enter the items that will appear in your radio buttons. For more information, see *How to Enter Items Manually in the List Source Dialog Box* on page 173.
   - **From a data source stack column.** Select *From a variable* and select the name of a data source stack column. You can create a data source stack by clicking *New data source stack*. For more information, see *How to Create a Data Source Stack Explicitly Using the Stack Editor* on page 73.
   - **From a variable.** Select *From a variable* and select the name of a variable. You can create a variable by clicking *New variable*. For more information, see *How to Create a Variable in a Procedure* on page 68.
     - **Note:** You must create a variable with the data type Stack of.
   - **From the ACCEPT list of a data source field.** Select *From a variable* and select the column in the list of variables. You will only see this option if the data source this procedure is using contains a field with an ACCEPT value.
7. Click OK.
8. If necessary, resize the radio button control on the form to display all of your values.
9. Select the `SelectedItem` property to open the Binding the Selection Result dialog box. Use this dialog box to bind the end user’s radio button choice to a data source stack column or variable.

**Caution:** Do not bind `SelectedItem` to a data source stack that already contains data. This will replace the contents of the data source stack with the result of the selection.

10. Select a data source stack column or variable. You can create a data source stack or variable by clicking `New data source stack` or `New variable`. For more information, see *How to Create a Data Source Stack Explicitly Using the Stack Editor* on page 73 or *How to Create a Variable in a Procedure* on page 68.

11. Determine whether to store the text or the value you entered in the List Source dialog box.

12. Click OK.

**Syntax:** **How to Set the Value of a Radio Button Group Dynamically**

If you want to set the value of a group of radio buttons dynamically, issue the following command

```
COMPUTE Formname.RadioButtonName.ListItems.FocIndex = n;
```

where:

- **Formname**
  
  Is the name of the form on which the radio button control is placed.

- **RadioButtonName**
  
  Is the name of the radio button control.

- **ListItems**
  
  Is the name of an internal data source stack that contains the values for the group of radio buttons.

You can also reset the values of all controls in a form to their initial values using the Winform Reset command. For more information, see *How to Reset a Form* on page 121.

**Procedure:** **How to Trigger an Action When an End User Selects an Item in a Radio Button Group**

1. Open the Event Handler editor.
2. Select the radio button in the list of controls.
3. Select the `Change` event from the list of events.
4. Specify an event handler.
Changing Radio Button Properties

When you select your radio buttons, you will see a list of radio button properties in the property sheet. Changing these properties will change what your radio buttons look like and what they do at run time.

Many of the styling properties can also be changed using a Cascading Style Sheet. For more information on Cascading Style Sheets, see Using Cascading Style Sheets on page 113.

Choose your property based on the task you wish to perform:

**If you want to change the contents of the radio buttons**, use the ListItems property to open the *List Source Dialog Box* on page 176.

**If you want to change the name of the radio button group that identifies it to the procedure**, use the *(Name)* Property. For more information, see the *(Name) Property* on page 267.

**If you want to assign the value selected by the end user to a variable**, use the SelectedItem property to open the *Binding the Selection Result Dialog Box* on page 179.

**If you want to change the size of the location or size of the radio buttons**, use a style sheet or resize the radio button group directly in the form. You can also use the *Bottom Property* on page 273, *Left Property* on page 286, *Right Property* on page 289, and *Top Property* on page 293.

**If you want to change the color of the radio buttons**, use a style sheet or the *ForeColor Property* on page 279 to determine the color of the text. Use the *BackColor Property* on page 269 to determine the color of the box.

**If you want to change the font in the radio buttons**, use a style sheet or use the *Font Property* on page 279.

**If you want to add a label to the radio buttons**, use the *Border Property* on page 271 set to anything but 0-None and the *BorderText Property* on page 272.

**If you want to add a border to the radio buttons**, use a style sheet, the *Border Property* on page 271 to determine whether you have a border, the *BorderColor Property* on page 272 to determine the color of the border, and the *BorderWidth Property* on page 273 to determine the width of the border.

**If you want to change the number of the columns or rows displayed**, use the Columns or Rows properties. For more information, see *Determining the Layout of Your Radio Buttons* on page 250.

**Note:** The size of the radio button group also determines how many rows or columns you see.
If you want to make the radio buttons inactive or make it invisible, use the Enabled Property to determine whether the radio buttons are active or not. If the radio button is inactive, nothing will happen when the end user clicks it. You can also use the Visible Property to determine whether the radio button is visible to the end user or use a style sheet. For more information, see the Enabled Property on page 278 and the Visible Property on page 294.

If you want to change what the cursor looks like when it is on top of the radio buttons, use the CursorPointer Property on page 276 or use a style sheet.

If you want to display a tooltip when the cursor is on the top of the radio buttons, use the ToolTipText Property on page 293.

If you want to assign a Help topic to the radio button, use the Help Property. For more information, see Assigning Help to Your Forms and Controls on page 263.

If you want the end user to be able to tab to the radio buttons, use the Tabstop Property on page 291.

If you want to move the radio button to another layer, use the Layer Property. For more information, see Layering Controls on page 103.

If you want to control the display order of the radio button when more than one control is overlaid, use the ZIndex Property. For more information, see the ZIndex Property on page 294.

Determining the Layout of Your Radio Buttons

There are three factors that determine the layout of the radio buttons in your radio button control:

- The size of the control determines the total area available for displaying radio buttons.
- The Columns property determines how many columns of buttons there are. The default is 1, which means that there is only one column. If you set Columns to -1, WebFOCUS Maintain will use as many columns as necessary.
- The Rows property determines how many rows of buttons there are. The default is -1, which means as many rows as necessary.
The order in which the buttons are laid out is from top to bottom, and then left to right, as shown in the following image.

For example, for the following radio button control, Columns has been set to 2, Rows has been set to -1, and the size has been adjusted so that the columns are as balanced as possible.

If you are not careful, some of your radio buttons may not be visible, either because your control is not large enough to handle all of the radio buttons or Rows and Columns are set improperly.

For example, if you set Rows to 3 and Columns to 2 for this radio button control, you will only see six radio buttons, and Sunday will disappear.

To solve this problem, we recommend that if you set Rows or Columns to a positive number, you set the other property to -1 so that you won’t accidentally lose any radio buttons.

You must also make sure that your control is large enough to display all of the radio buttons. The Form Editor displays the number of radio buttons that will be displayed at run time.

If your radio button control will have a variable number of radio buttons, we recommend that you consider using a list box control or combo box control instead, so that you can better control the layout of controls on your form.

**Using Text**

The text control enables you to add static text areas to a form.
**Procedure:** How to Place Text on Your Form

1. Select the Text control in the Controls palette.
2. Draw a rectangle on your form approximately where you want your text block to be at approximately the size you want.
3. Type the text you want your button to have and press Enter. (The text should be selected automatically when you created the button.) To move to the next line, press Shift+Enter.
4. If necessary, readjust the size and placement of your text.

**Changing Text Properties**

When you select your text, you will see a list of text properties in the property sheet. Changing these properties will change what your text looks like.

Many of the styling properties can also be changed using a Cascading Style Sheet. For more information on Cascading Style Sheets, see *Using Cascading Style Sheets* on page 113.

Choose your property based on the task you wish to perform:

**If you want to change the text,** use the *Text Property* on page 292. You can also select the text and change it directly on the form.

**If you want to change the name of the text that identifies it to the procedure,** use the *(Name)* Property. For more information, see the *(Name)* Property on page 267.

**If you want to change the size of the location or size of the text control,** use a style sheet or resize the text directly in the form. You can also use the *Bottom Property* on page 273, *Left Property* on page 286, *Right Property* on page 289, and *Top Property* on page 293.

**If you want to change the color of the text,** use a style sheet or the *ForeColor Property* on page 279 to determine the color of the text. If your text control has an event handler assigned to its Click event, the Hyperlink property uses the browser settings of the end user to determine the color of the text. Use the *BackColor Property* on page 269 to determine the background color.

**If you want to change the text font,** use a style sheet or use the *Font Property* on page 279.

**If you want to change the text alignment to left-justified, centered, or right-justified,** use a style sheet or the *Alignment Property* on page 267.

**If you want to display a tooltip when the cursor is on the top of the text,** use the *ToolTipText Property* on page 293.

**If you want to assign a Help topic to the text,** use the Help Property. For more information, see *Assigning Help to Your Forms and Controls* on page 263.
If you want to add a border to the text, use a style sheet, the Border Property on page 271 to determine whether you have a border, the BorderColor Property on page 272 to determine the color of the border, and the BorderWidth Property on page 273 to determine the width of the border.

If you want to make the text inactive or make it invisible, use the Enabled Property to determine whether the text are active or not. If the text is inactive, nothing will happen when the end user clicks it. You can also use the Visible Property to determine whether the text is visible to the end user or use a style sheet. For more information, see the Enabled Property on page 278 and the Visible Property on page 294.

If you want to change what the cursor looks like when it is on top of the text, use the CursorPointer Property on page 276 or use a style sheet.

If you want to move the text to another layer, use the Layer Property. For more information, see Layering Controls on page 103.

If you want to control the display order of the text when more than one control is overlaid, use the ZIndex Property. For more information, see the ZIndex Property on page 294.

Using Text Controls as Hyperlinks

One common use for text controls is in hyperlinks. Usually, hyperlinks are distinguished from other text by being underlined and a different color. Also, when you move your cursor on top of them, it turns into a hand.

**Procedure:** How to Turn Your Text Control Into a Hyperlink

1. Open the Event Handler editor for your text control and select the Click event.
2. Assign an event handler to your control (usually, a web link).
3. Close the Event Handler editor.
4. Change the Hyperlink property to 1 - Yes.

**Note:** You must have an event handler assigned to the Click event for the text control, or the hyperlink will not show up.

Using ActiveX Controls

You can use ActiveX controls in your forms. ActiveX controls are compiled software components that provide a set of business or user interface functions. They can extend the functionality of your applications.
ActiveX controls are included in your project as resources. Resources are part of your project, but you do not create or edit them directly using the Maintain Development Environment. However, WebFOCUS Maintain will take care of deploying them to the correct location so that your application will run properly.

When you place an ActiveX control on your form, the property sheet for that control gains an extra tab: the ActiveX tab. The properties in this tab determine the behavior of your ActiveX control. Use JavaScript or VBScript for your Maintain procedure to access these properties.

Depending on the security settings of your end users browsers, ActiveX controls may be disabled. For example, to change your security settings for Internet Explorer 5, in the Tools menu, click Internet Options, then click the Security tab, and click Custom Level. You will see a Security Settings dialog box where you can enable or disable various features, including ActiveX controls.

**Procedure: How to Place an ActiveX Control on Your Form**

1. Select the ActiveX control in the Controls palette.
2. Draw a rectangle on your form where you want your ActiveX control to be at approximately the size you want.
3. If you have no ActiveX controls as resources in your project, WebFOCUS Maintain opens the Resource Wizard, where you can add one as a resource. For more information, see How to Add an ActiveX Control to Your Project as a Resource on page 254. When you finish adding your ActiveX control to the project, WebFOCUS Maintain will display it in the Insert ActiveX Control Resource dialog box.

   If you have ActiveX controls as resources in your project, WebFOCUS Maintain will display the Insert ActiveX Control Resource dialog box.

4. Select the ActiveX control you want to use and click OK.
5. Giving you ActiveX control a more meaningful name than the default one is optional, but recommended.
6. If necessary, readjust the size and placement of your ActiveX control.
7. To change any properties specific to the ActiveX control, click the ActiveX tab in the property sheet.

**Procedure: How to Add an ActiveX Control to Your Project as a Resource**

1. If you do not have ActiveX controls in your project, use the ActiveX control in the Form Editor to add an ActiveX control to your form.

   or
In the Insert ActiveX Control Resource dialog box, click Create a new ActiveX control resource. WebFOCUS Maintain opens the Resource Wizard, a series of windows that guide you through adding an ActiveX control to your project.

2. Click Browse for registered controls on My Computer.
3. Select an ActiveX control in the Insert ActiveX Control dialog box and click OK.
   You will return to the Resource Wizard with the ActiveX control name and location listed.
4. Click Next.
5. Specify a unique name for your ActiveX control, then click Finish.
6. You may see a dialog box informing you that WebFOCUS Maintain needs to make a copy of the ActiveX control in the project directory. Click OK.

**Syntax:** How to Pass the Value of an ActiveX Control Property to a Maintain Function

To use the value of an ActiveX Control property in a Maintain function, use JavaScript or VBScript to retrieve it. In an event handler for an ActiveX control event, use a script function for the handler and then call the Maintain function using IWCTrigger. For more information on IWCTrigger, see How to Use the IWCTrigger Function to Call a Maintain Function From Your Script Handler on page 137.

IWCTrigger ("MaintainFunction",document.form.control.property);

where:

MaintainFunction
   Is the name of the Maintain function you are calling.

form
   Is the name of the form the ActiveX control is on.

control
   Is the name of the ActiveX control.

property
   Is the name of the ActiveX control property (look for ActiveX control properties in the ActiveX tab of the property sheet for the ActiveX control).
**Example:** Passing an ActiveX Control Value to a Maintain Function

If you have an ActiveX calendar control on Form1 which has a property called Month, you can use IWCTrigger to send the value of Month to a Maintain function called UpdateDate, using either JavaScript or VBScript:

IWCTrigger("UpdateDate",document.Form1.CalendarControl.Month);

**Reference:** Insert ActiveX Control Dialog Box

Use the Insert ActiveX Control dialog box, shown in the following image, to select an ActiveX control to include as a resource in your project.

![Insert ActiveX Control Dialog Box](image)

This dialog box contains the following elements:

**Select a control type:**

Contains a list of the supported ActiveX controls available on your machine.

**Refresh**

Tells WebFOCUS Maintain to update the list of ActiveX controls when you add a new one to your machine.
Reference: Insert ActiveX Control Resource

Use the Insert ActiveX Control Resource dialog box to select which ActiveX resource to insert into your form.

This dialog box has the following elements:

Select an ActiveX control resource

Lists the available ActiveX control resources in your project.

Create a new ActiveX control resource

Opens the Resource Wizard, where you can add a new ActiveX control to your project.

Changing ActiveX Control Properties

When you select your ActiveX control, you will see a list of ActiveX control properties in the property sheet. Changing these properties will change what your ActiveX control looks like and what it does at run time.

Many of the styling properties can also be changed using a Cascading Style Sheet. For more information on Cascading Style Sheets, see Using Cascading Style Sheets on page 113.
An ActiveX control also contains an ActiveX tab in the property sheet that contains more properties that are specific to the control. For more information on these properties, see the documentation that accompanied the control.

Choose your property based on the task you wish to perform:

**If you want to change the name of the ActiveX control that identifies it to the procedure,** use the (Name) Property. For more information, see the *(Name) Property* on page 267.

**If you want to change the size of the location or size of the ActiveX control,** use a style sheet or resize the ActiveX directly in the form. You can also use the *Bottom Property* on page 273, *Left Property* on page 286, *Right Property* on page 289, and *Top Property* on page 293.

**If you want to make the ActiveX control inactive or make it invisible,** use the Enabled Property to determine whether the ActiveX control is active or not. If the ActiveX control is inactive, nothing will happen when the end user clicks it. You can also use the Visible Property to determine whether the ActiveX control is visible to the end user or use a style sheet. For more information, see the *Enabled Property* on page 278 and the *Visible Property* on page 294.

**If you want to change what the cursor looks like when it is on top of the ActiveX control,** use the *CursorPointer Property* on page 276 or use a style sheet.

**If you want to display a tooltip when the cursor is on the top of the ActiveX control,** use the *ToolTipText Property* on page 293.

**If you want to assign a Help topic to the ActiveX control,** use the *Help Property*. For more information, see *Assigning Help to Your Forms and Controls* on page 263.

**If you want the end user to be able to tab to the ActiveX control,** use the *Tabstop Property* on page 291.

**If you want to move the text to another layer,** use the Layer Property. For more information, see *Layering Controls* on page 103.

### Unstable ActiveX Controls

Occasionally, you may have unstable ActiveX controls on your computer that have been installed with other software or downloaded from the web. If your computer has any unstable ActiveX controls when you first generate the list of available controls for the Insert ActiveX control dialog box, or when you refresh the list, a control may cause the Maintain Development Environment to crash while it is reading the control. However, from that point on, WebFOCUS Maintain will keep a list of any unstable ActiveX controls in the registry and will no longer try to load them into the list. Therefore, you will never crash more than once over the same unstable control.
**Note:** The only way to take a control off of the black list is to delete it from the registry.

**Dynamically Manipulating Controls at Run Time**

You manipulate controls at run time using the Winform command. For more information, see *How to Determine the Value of a Control Property* on page 259 and *How to Set the Value of a Control Property* on page 259.

**Syntax:** **How to Determine the Value of a Control Property**

To determine the value of a control property and assign it to a variable, use the following syntax

```
Winform Get form.control.property INTO variable ;
```

or

```
variable = form.control.property ;
```

where:

- `form` is the name of the form, determined by the (Name) property.
- `control` is the name of the control, determined by the (Name) property.
- `property` is the name of the property.

**Syntax:** **How to Set the Value of a Control Property**

To set the value of a control property (except for colors), use the following syntax

```
[Winform Show_Inactive form;]
Winform Set form.control.property TO setting ;
```

or

```
[Winform Show_Inactive form;]form.control.property = setting ;
```

where:

- `Winform Show_Inactive` is required if the form is not the active form.
Defining Colors for Your Form and Controls

form

Is the name of the form, determined by the (Name) property.

control

Is the name of the control, determined by the (Name) property.

property

Is the name of the property.

Not all properties can be set dynamically. To see which properties you can set, see Form and Control Properties Reference on page 267.

Tip: You can generate the Winform Set syntax by selecting the property in the property sheet and dragging it to the Procedure Editor or the Event Handler editor. If you cannot drag the property into the Procedure Editor or the Event Handler editor, then the property is not dynamically "settable."

For more information on how to set color properties, see Defining Colors for Your Form and Controls on page 260.

Defining Colors for Your Form and Controls

Several control properties enable you to change the color of various aspects of your control, such as the background color, border color, text color, row color (in HTML Tables), and so on. Use the Color well to assign colors to all of these properties.

The following is a list of properties that assign colors. For more information on these properties, see Form and Control Properties Reference on page 267.

- AlternateRowColor makes alternate rows in an HTML Table or Grid different colors.
- BackColor determines the color of the form or control.
- BackColorOver determines the color of the background when the cursor is on top of a menu.
- BorderColor determines the color of borders in a control.
- ForeColor determines the color of text in a control.
- ForeColorOver determines the color of the text when the cursor is on top of a menu.
- HeaderBackColor determines the color of the header row of an HTML Table or Grid.
- HeaderForeColor determines the color of the text in the header row of an HTML Table or Grid.
If a control has any of these attributes set to Default, then it uses the color settings for BackColor and ForeColor for the form itself, unless it is one of the following:

- The Grid.
- An input style control, such as a Button, EditBox, MultiEditBox, ComboBox or ListBox. Input style controls default to the CSS settings if a Style Sheet is used; otherwise use the BackColor and ForeColor properties to change the colors of an individual control.

You can assign one of the twenty colors in the Colors box, or you can click Other to open the Colors dialog box, where you can assign one of forty-eight predefined colors or define your own custom color.

Note: You can also assign colors using a Cascading Style Sheet. If your form has a Cascading Style Sheet, setting any color property to Default means that WebFOCUS Maintain will look in the style sheet for the color to use. If you set any color property to anything other than Default, then WebFOCUS Maintain will use that setting. For more information on Cascading Style Sheets, see Using Cascading Style Sheets on page 113.

**Procedure: How to Add Colors**

1. Select the control or form.
2. Click the button on the left of the selected property box in the property sheet.
   
   The Color drop-down list box opens.
3. Select one of the colors in the box.
   
   or
4. Click OK to apply the color.

**Procedure: How to Define Custom Colors**

1. Click Other in the Color box.
2. Select one of the basic colors from the Basic Color chart.
3. Click or drag the arrow on the side of the Color Box to change the current color displayed in the Color|Solid box.
   
   or
Defining Colors for Your Form and Controls

Type the desired values in the boxes below the Color Box. The boxes on the left use hue, saturation, and luminosity. The boxes on the right use the three primary colors used by color monitors: red, green, and blue.

4. Click the Add to Custom Colors button.

The color is added to the Custom Colors chart on the left.

5. Click the desired color from the Custom Colors chart. A dotted line borders the chosen color.

6. Click OK to apply the color.

For more information on using the Colors dialog box, click the question mark next to the close box and then click on any element in the dialog box.

**Syntax:** How to Set the Value of a Control Color Property

To set the value of a form or control color property, use the following syntax

```
[Winform Show_Inactive form;]form.[control.]SETcolorproperty(amount_of_red, amount_of_green, amount_of_blue);
```

where:

- **Winform Show_Inactive**
  
  Is required if the form is not currently the active form.

- **form**
  
  Is the name of the form, determined by the (Name) property.

- **control**
  
  Is the name of the control, determined by the (Name) property.

- **colorproperty**
  
  Is the name of the color property (BackColor, ForeColor, and so on).

- **amount_of_red, amount_of_green, amount_of_blue**
  
  Defines the color by the amount of red, green, and blue (RGB) present in the color. For more information on how these colors work, open the Colors dialog box, click the question mark next to the close box, and then click Red, Green, or Blue.

**Note:** You can generate this syntax by selecting the property in the property sheet and dragging it to the Procedure Editor or the Event Handler editor.
Assigning Help to Your Forms and Controls

All WebFOCUS Maintain developers would like to write applications that are intuitively obvious to even the most casual observer. Occasionally, however, end users need a little nudge in the right direction.

Using the Help property, you can assign a web link to a form or control. If your web link is an HTML file, then you can also assign an anchor that identifies a location in the HTML file. Then, when the end user presses the F1 key in the form or while focus is on the control, the web link opens.

Procedure: How to Assign Help to Your Forms and Controls

1. Select the form or control.
2. Select the Help property to open the Help dialog box.
3. Select the name of a web link that is an HTML file from the list of web links in your project.
   or
   Click New to define a new web link. For more information on defining a web link, see How to Create a Web Link on page 144.
4. Optionally, enter the name of an anchor in the HTML file so that your application will jump to the location of the anchor instead of the beginning of the HTML file.
5. Click OK when you are done.

Reference: Help Dialog Box

Use the Help dialog box, shown in the following image, to determine the name of a web link, associated with an HTML file, to open when end users press F1 with their focus on this control.
Assigning Tab Order to Controls

This dialog box has the following elements:

**List of web links in your project**

Lists the web links that you have defined in this project.

**New**

Opens the Link Editor so that you can define a new web link. For more information, see *How to Create a Web Link* on page 144.

**Optionally, specify a help anchor**

Determines a location in an HTML file to open at, instead of opening it at the beginning.

**Assigning Tab Order to Controls**

WebFOCUS Maintain automatically assigns tab order according to the sequence of the controls placed on your form. For example, if you place two buttons on a form, the tab order of the first button placed on the form would be one, and the tab order of the second button would be two. However, you can use the Tab Order dialog box to assign your own tab order to controls.

Tab order cannot be assigned to text, group boxes, lines, or controls that have the Tabstop property turned off.

Using Move to Front or Move to Back also affects the tab order.

**Note:** Tab order overrides the DefaultButton property, so if you have a button with the DefaultButton property turned on, make sure you do not have another button before it in the tab order.

**Procedure:** How to Assign Tab Order to Controls

1. Click the Tab order button on the Layout toolbar.

   or

   In the Layout menu, click Tab order. The Tab Order window opens, displaying a list of all the controls on your form.

2. To move the tab order of a control up, select it and click the Move up button.

   To move the tab order of a control down, select it and click the Move down button.

3. Click OK to save your changes.
The list will reflect the tab order of the controls on your form. At run time, the end user will be able to tab through the fields in your form in the order you assigned.

**Reference:**  Tab Order Dialog Box

Use the Tab Order dialog box, shown in the following image, to determine in what order an end user pressing Tab in your form will go.

![Tab Order Dialog Box](image)

This dialog box includes the following options:

**List of controls on form**

Lists the controls that you have placed on the form, along with a description of what type they are. Notice that you cannot tab to text, an image, or a control that has the Tabstop property turned off, so these controls are not assigned an order.

Moves the selected control up in the list.

Moves the selected control down in the list.
You determine the appearance and behavior of the forms and controls in your application by setting their properties. Following is an alphabetical list of all of the form and control properties.

You can set some of these properties dynamically at run time using the Winform Set command. You can also obtain the value of some of these properties using the Winform Get command.

**GroupName** Property

The (GroupName) property is the name the Form Editor assigns to a collection of grouped controls. You cannot change the value for this property. For more information on grouped controls, see Grouping Controls on page 101.

**Applies to:** Grouped controls.

**Name** Property

The (Name) property is the name of the control or form as it is known to your procedure. Your end users do not need to know what this name is. The (Name) is different from the title or text that appears on the control, which the end user does see.

**Note:** It is recommended that you give your forms and controls useful names instead of the default (for example, a button with the text Add that inserts data into a data source might be called AddButton).

**Applies to:** Forms, all controls.

**Alignment Property**

The Alignment property determines whether the text in controls is centered, left-justified, or right-justified.

You can also set this property using a cascading style sheet. For more information, see Using Cascading Style Sheets on page 113.

**Applies to:** Edit boxes, group boxes, multi-edit boxes, text.
**Setting dynamically:**

Winform Set form.control.ALIGNMENT TO n;

**Property Settings:**

0 - Left

Aligns text to the left in the selected control (default for edit box, group box, and multi-edit box).

1 - Center

Centers text in the selected control (default for text control).

2 - Right

Aligns text to the right in the selected control.

3 - AlignByType

(Edit box only) Left-justifies character input, and right-justifies numeric input.

**Alt Property**

The Alt property enables you to display pop-up information about an image when end users place their cursors over the image.

Alt is different from ToolTipText in that Alt is read by all screen readers for the blind.

**Applies to:** Images.

**Setting dynamically:**

Winform Set form.[control.]ALT TO "text";

**AlternateRowColor Property**

The AlternateRowColor property enables you to make alternate rows in an HTML Table different colors (useful for "ledger-formatted" output). For example, you could set up your table to have alternating white and green rows. For more information, see Defining Colors for Your Form and Controls on page 260.

**Note:** Avoid using a dithered background color since text is often not as readable when displayed on top of it.

For HTMLTable only, you can also set this property using a cascading style sheet. For more information, see Using Cascading Style Sheets on page 113.

**Applies to:** Grids and HTML Tables.
Setting dynamically:

`form.control.SETALTERNATEROWCOLOR(amount_of_red, amount_of_green, amount_of_blue);`

**Property Settings:**

**Default**

Sets the alternate row color to be the same as BackColor (which means you have no alternate row color), if you do not have a cascading style sheet applied to your form. Otherwise, it uses the setting in the cascading style sheet.

`amount_of_red, amount_of_green, amount_of_blue`

Sets the alternate row to be the color defined by the amount of red, green, and blue.

**BackColor Property**

The BackColor property specifies the background color of the form or control. For more information, see *Defining Colors for Your Form and Controls* on page 260.

**Note:** Avoid using a dithered background color, since text is often not as readable when displayed on top of it.

You can also set this property using a cascading style sheet, except for the Grid. For more information, see *Using Cascading Style Sheets* on page 113.

**Applies to:** Grids, forms, buttons, check boxes, combo boxes, edit boxes, HTML Objects, HTML Tables, list boxes, menus, multi-edit boxes, radio buttons, text.

**Setting dynamically:**

`form.[control.]SETBACKCOLOR(amount_of_red, amount_of_green, amount_of_blue);`

**Property Settings:**

**Default**

Sets gray as the background color, if you do not have a cascading style sheet applied to your form. Otherwise, it uses the setting in the cascading style sheet.

`amount_of_red, amount_of_green, amount_of_blue`

Sets the background to be the color defined by the amount of red, green, and blue.
BackColorOver Property

The BackColorOver property determines the color of the background when the cursor is on top of a menu. For more information, see Defining Colors for Your Form and Controls on page 260.

**Note:** Avoid using a dithered background color, since text is often not as readable when displayed on top of it.

You can also set this property using a cascading style sheet. For more information, see Using Cascading Style Sheets on page 113.

**Applies to:** Menus.

**Setting dynamically:**

```sql
form.menu.]SETBACKCOLOROVER(amount_of_red, amount_of_green, amount_of_blue);
```

**Property Settings:**

- **Default**
  
  Sets gray as the background color, if you do not have a cascading style sheet applied to your form. Otherwise, it uses the setting in the cascading style sheet.

  ```sql
  amount_of_red, amount_of_green, amount_of_blue
  ```

  Sets the background to be the color defined by the amount of red, green, and blue.

BackgroundImage Property

The BackgroundImage property specifies an image as a background for a form. To set the value for this property, you use the Image Source dialog box. For more information, see Using Images on page 223.

You can also set this property using a cascading style sheet. For more information, see Using Cascading Style Sheets on page 113.

**Setting dynamically:** Yes, you can set its value to a Maintain variable.

**Applies to:** Forms.

BodyRowHeight Property

The BodyRowHeight property controls the height of the rows in the body of an HTML Table control.

**Applies to:** HTML Tables.

**Property Settings:** Enter a numeric value for the row height in the body of an HTML Table.
Border Property

The Border property specifies the type of border that appears on a control.

You can also set this property using a cascading style sheet. For more information, see Using Cascading Style Sheets on page 113.

Applies to: Edit boxes, frames, group boxes, HTML Objects, HTML Tables, images, Java applets, menus, multi-edit boxes, radio buttons, and text.

Setting dynamically:

Wiform Set form.control.BORDER TO n;

Property Settings for edit box, multi-edit box, menu, frame:

0 - No
Displays no border.

1 - Yes
Displays a border (default for menu).

Default
Displays border according to cascading style sheet (CSS) settings. If nothing is set, the default settings of the browser are used (default for edit box, multi-edit box, frame. Not available for menu).

Property Settings for HTML Object, HTML Table, image, radio button, text, group box, Java applet:

0 - None
Displays no border.

1 - Normal
Displays a flat border.

2 - 3D
Displays a border with a 3-dimensional effect (default for HTML Table).

3 - Sunken
Displays a shadow on the upper left side of the border that appears to be etched into the form.
BorderColor Property

4 - Raised
Displays a shadow on the lower right side of the border that appears to be etched into the form (not available for group box).

Default
Displays border according to css settings. If nothing is set, the default settings of the browser are used (default for HTML Object, image, radio button, text, group box, Java applet. Not available for HTML Table).

BorderColor Property

TheBorderColor property enables you to select a color for the border of a control. For more information, see Defining Colors for Your Form and Controls on page 260.

You can also set this property using a cascading style sheet. For more information, see Using Cascading Style Sheets on page 113.

Applies to: Group boxes, HTML Objects, HTML Tables, images, radio buttons, text.

Setting dynamically:

form.control.SETBORDERCOLOR(amount_of_red, amount_of_green, amount_of_blue);

Property Settings:

Default
Sets black as the border color, if you do not have a cascading style sheet applied to your form. Otherwise, it uses the setting in the cascading style sheet.

amount_of_red, amount_of_green, amount_of_blue
Sets the border to be the color defined by the amount of red, green, and blue.

BorderText Property

The BorderText property places a label in the border of a radio button. In order for this label to show up, you must have a border. You can use this property to describe a set of radio buttons.

Applies to: Radio buttons.

Setting dynamically:

Winform Set form.radiobutton.BORDERTEXT TO "text";
BorderWidth Property

The BorderWidth property determines the border width of a control, measured in pixels.

You can also set this property using a cascading style sheet. For more information, see Using Cascading Style Sheets on page 113.

**Applies to:** Group boxes, HTML Objects, HTML Tables, images, radio buttons, text.

**Setting dynamically:**

```
Winform Set form.control.BORDERWIDTH TO n;
```

Bottom Property

The Bottom property determines where the bottom of a control is on the form, measured in pixels. Together with the Top, Left, and Right properties, it determines the size of your control and where it is on the form.

You can also set this property using a cascading style sheet. For more information, see Using Cascading Style Sheets on page 113.

**Applies to:** All controls.

**Setting dynamically:**

```
Winform Set form.control.BOTTOM TO n;
```

Calendar Property

The Calendar property provides a calendar icon at run time for any date-formatted edit box on a form. By clicking the icon, a pop-up interactive calendar is then made available to users so that they can select date values, rather than having to type them in.

**Applies to:** Forms, edit boxes.

**Setting dynamically:** No.

**Property Settings:**

- **0 - No**
  - Disables the Calendar property. This is the default value for Forms.

- **1 - Yes**
  - Enables the Calendar property. This will automatically provide a Calendar icon on the form for any edit box using a date format.
2 - TimeOnTop

Sets the display of the date-time values at the top of the pop-up calendar. By default, the
date-time values are displayed at the bottom of the pop-up calendar. This is a Form property
only.

The default setting for an edit box is Yes. The edit box setting works in conjunction with the Form
Calendar property setting as follows:

- When the Form Calendar property is Yes, the Calendar property for individual edit boxes can
  be set to No to prevent the display of the calendar icon at run time for those particular controls.

- When the Form Calendar property is set to No, the pop-up calendar will be disabled for all
date-format edit boxes on the form and is not enabled for individual controls when the edit
box Calendar property is set to Yes.

The javascripts and default stylesheet used by the Calendar icon are included in the
\ibi_html\javaassist\ibi\html\maint directory of both Developer Studio and the WebFOCUS Client
and are available to the application at run time.

CaseStyle Property

The CaseStyle property enables you to convert input in an edit box or multi-edit box to uppercase
or lowercase.

**Applies to:** Edit boxes, multi-edit boxes.

**Setting dynamically:**

Winform Set form.control.CASESTYLE TO n;

**Property Settings:**

0 - Unchanged

- Displays input as it was originally entered, whether uppercase, lowercase, or mixed-case
  (default).

1 - Lower

- Converts all input to lowercase.

2 - Upper

- Converts all input to uppercase.
Checked Property

The Checked property determines if a check box is initially selected when an end user opens a form. It also indicates whether the end user selected or cleared the check box during run time. You set this value using the Set Check Box State dialog box. For more information, see the Set Check Box State Dialog Box on page 169.

**Applies to:** Check boxes.

**Setting dynamically:** Yes, you can set its value to a Maintain variable.

Columns Property

For the HTML Table and Grid controls, the Columns property determines which columns of the data source stack you are viewing will appear and in what order. You set the columns using the Control Columns dialog box. For more information, see Control Columns Dialog Box on page 195.

For the radio button control, the Columns property determines how many columns of radio buttons to draw. The value -1 indicates that WebFOCUS Maintain will place as many columns as can fit in the box. You can also enter any number greater than 0. Together with the Rows property, this property determines the layout of your radio buttons. For more information, see Determining the Layout of Your Radio Buttons on page 250.

**Applies to:** Grids, HTML Tables, radio buttons.

**Setting dynamically for Grids and HTML Tables:** No.

**Setting dynamically for radio buttons:**

Winform Set form.control.COLUMNS TO n;

Content Property

The Content property determines the HTML code to be inserted inline on the form. You set its value using the HTML Content Source dialog box. For more information, see HTML Content Source Dialog Box on page 220.

**Applies to:** HTML Objects.

**Setting dynamically:** Yes, you can set its value to a Maintain variable.

CSSName Property

The CSSName property sets a stylesheet for the form.

**Applies to:** Forms.
CSSClass Property

The CSSClass property points to a specific class in a CSS file. You can use individual CSS classes to set fonts and colors for forms as well as groups of controls. Use with the CSSName property.

**Applies to:** Forms and all controls except grids and menu.

**Setting dynamically:**

```plaintext
Winfom Show_inactive form
Winfom Set form.CSSClass TO 'classname';
Winfom Show form
```

**Property Settings:**

Enter the name of the CSS class within a CSS file, and provide the CSS file name in the CSSName property. The CSS file must be included in the project during the development process and assigned to a web server when deployed.

For more information, see the CSSName property.

CursorPointer Property

The CursorPointer property specifies the type of cursor displayed when an end user moves the cursor over a form or control. For example, it is customary for your cursor to turn into a hand when you move it over a link or a button.

You can also set this property using a cascading style sheet. For more information, see *Using Cascading Style Sheets* on page 113.

**Applies to:** Forms, all controls except for combo boxes, frames, and list boxes.

**Setting dynamically:**

```plaintext
Winfom Set form.[control.]CURSORPOINTER TO n;
```
**Property Settings:**

0 - Default

Displays an arrow (↑) when end users move their cursors over the control, if you do not have a cascading style sheet applied to your form. Otherwise, it uses the setting in the cascading style sheet.

1 - IBeam

Displays a text-editing I-beam (↑) when end users move their cursors over the control. This cursor is useful to indicate that end users can enter text into a control, such as an edit box or multi-edit box.

2 - Move

Displays a move cursor (←→).

3 - Cross

Displays a cursor shaped like a cross (↓↑).

4 - Wait

Displays an hourglass cursor (⌛). This cursor indicates that end users must wait until your application finishes a task.

5 - Help

Displays a cursor with a question mark (❓). This cursor is useful to indicate that end users can get help.

6 - Hand

Displays a hand cursor (👉). This cursor is useful to indicate that end users can accomplish something by clicking the control, such as a web link or button.

**DefaultButton Property**

The DefaultButton property enables you to make the selected button control the default when the end user presses the Enter key.
**Note:** Make sure that you do not have another button first in the tab order. The tab order will override the DefaultButton property.

**Applies to:** Buttons.

**Setting dynamically:**

Winform Set `form.button.DEFAULTBUTTON` TO `{0|1};`

**Property Settings:**

0 – No

Disables the default button feature (the default).

1 – Yes

Enables the default button feature.

---

**Enabled Property**

The Enabled property specifies whether a control can respond to user generated events. Disabled controls are not part of the tab order.

**Applies to:** All controls, except for frames.

**Setting dynamically:**

Winform Set `form.control.ENABLED` TO `{0|1};`

**Property Settings:**

0 – No

Turns off the enable feature and disables the control.

1 – Yes

Turns on the enable feature (the default).

---

**FixedColumns Property**

The FixedColumns property determines which columns remain stationary while scrolling through a grid.

**Applies to:** Grids.

**Setting dynamically:** No.
Font Property

The Font property enables you to determine the typeface, style, and size of text in controls on the form. You can either set the font using the Fonts dialog box, or using a cascading style sheet.

If you do not have a cascading style sheet applied to your form, you can set the font for the form, and then all controls on the form will inherit that font.

If you want to change the font of an individual control, use the Fonts dialog box to change the font for just that control.

The default form font is MS Sans Serif, Regular, 8 point.

If you have a cascading style sheet applied to your form, you can set the font for elements in your form, other than the Grids, using the Style Sheet.

Make sure that the font is set to Default for every control, as well as the form itself (setting the font to anything other than Default takes precedence over the style sheet).

If a control has a class assigned to it, it will use the font defined for that class. If a control does not have a class assigned to it, it will use the default font defined in the style sheet.

For more information on Cascading Style Sheets, see Using Cascading Style Sheets on page 113.

**Applies to:** Grids, forms, buttons, check boxes, combo boxes, edit boxes, group boxes, HTML Objects, HTML Tables, list boxes, multi-edit boxes, radio buttons, text.

**Setting dynamically:** No.

ForeColor Property

The ForeColor property enables you to select the color for text of a control from the Color Palette. For more information, see Defining Colors for Your Form and Controls on page 260.

If you change the ForeColor property for the form itself, you set the value for ForeColor for all of the controls on the form that use the Default setting other than the following input style controls: Button, EditBox, and MultiEditBox. Cascading style sheets do not apply to the Grid.

You can also set this property using a cascading style sheet, except for the Grid. For more information, see Using Cascading Style Sheets on page 113.

**Applies to:** Grids, forms, buttons, check boxes, combo boxes, edit boxes, group boxes, HTML Objects, HTML Tables, lines, list boxes, menus, multi-edit boxes, radio buttons, text.

**Setting dynamically:**

```
form.[control.]SETFORECOLOR(amount_of_red, amount_of_green, amount_of_blue);
```
**Property Settings:**

**Default**

Sets the foreground color to be black, or if it has been set, the value of the form for ForeColor, if you do not have a cascading style sheet applied to your form. Otherwise, it uses the setting in the cascading style sheet.

*amount_of_red, amount_of_green, amount_of_blue*

Sets the control to be the color defined by the amount of red, green, and blue.

---

**ForeColorOver Property**

The ForeColorOver property determines the color of the text when the cursor is on top of a menu. For more information, see *Defining Colors for Your Form and Controls* on page 260.

You can also set this property using a cascading style sheet. For more information, see *Using Cascading Style Sheets* on page 113.

**Applies to:** Menus.

**Setting dynamically:**

```csharp
form.menu.]SETFORECOLOROVER(amount_of_red, amount_of_green, amount_of_blue);
```

---

**Property Settings:**

**Default**

Sets the foreground color to be black, or if it has been set, the value of the form for ForeColor, if you do not have a cascading style sheet applied to your form. Otherwise, it uses the setting in the cascading style sheet.

*amount_of_red, amount_of_green, amount_of_blue*

Sets the text to be the color defined by the amount of red, green, and blue.

---

**GridLines Property**

The GridLines property determines whether a grid or HTML Table displays grid lines.

**Applies to:** Grids, HTML Tables.

**Setting dynamically:**

```csharp
Winform Set form.controlGRIDLINES TO {0|1};
```
**Property settings:**

0 - No  
Hides the grid lines.

1 - Yes  
Displays the grid lines (the default).

**HeaderBackColor Property**

The HeaderBackColor property determines the back color for the header row of an HTML Table. For more information, see *Defining Colors for Your Form and Controls* on page 260.

**Tip:** Avoid using a dithered background color since text is often not as readable when displayed on top of it.

You can also set this property using a cascading style sheet, except for the Grid. For more information, see *Using Cascading Style Sheets* on page 113.

**Applies to:** Grids and HTML Tables.

**Setting dynamically:**

```csharp
form.control.SETHEADERBACKCOLOR(amount_of_red, amount_of_green, amount_of_blue);
```

**Property Settings:**

*Default*  
Sets gray as the header back color, if you do not have a cascading style sheet applied to your form. Otherwise, it uses the setting in the cascading style sheet.

*amount_of_red, amount_of_green, amount_of_blue*  
Sets the header to be the color defined by the amount of red, green, and blue.

**HeaderFont Property**

The HeaderFont property determines the font for the header row of an HTML Table.

You can also set this property using a cascading style sheet, except for the Grid. For more information, see *Using Cascading Style Sheets* on page 113.

**Applies to:** Grids and HTML Tables.

**Setting dynamically:** No.
**HeaderForeColor Property**

The **HeaderForeColor** property determines the color of the font in the header row of an HTML Table. For more information, see *Defining Colors for Your Form and Controls* on page 260.

You can also set this property using a cascading style sheet, except for the Grid. For more information, see *Using Cascading Style Sheets* on page 113.

**Applies to:** Grids and HTML Tables.

**Setting dynamically:**

```
form.control.SETHEADERFORECOLOR(amount_of_red, amount_of_green, amount_of_blue);
```

**Property Settings:**

*Default*

Sets the font color in the header to be black, or, if it has been set, the form's value for ForeColor, if you do not have a cascading style sheet applied to your form. Otherwise, it uses the setting in the cascading style sheet.

```
amount_of_red, amount_of_green, amount_of_blue
```

Sets the text in the header to be the color defined by the amount of red, green, and blue.

**Headers Property**

The **Headers** property determines whether your HTML Table has a header row. You can also remove or add the header row using the Control Columns dialog box. For more information, see *Control Columns Dialog Box* on page 195.

You can also set this property using a cascading style sheet, except for the Grid. For more information, see *Using Cascading Style Sheets* on page 113.

**Applies to:** Grids and HTML Tables.

**Setting dynamically:**

```
Winform Set form.control.HEADERS TO {0|1};
```

**Property Settings:**

*0 - No*

Does not display a header row.

*1 - Yes*

Displays a header row (the default).
**Help Property**

The Help property determines the name of web link associated with this control or form, and an optional location. When the end user presses F1 when the focus is on this form or control, WebFOCUS Maintain opens the link (either at the top or at the optional location).

To set the value for this property, you use the Help dialog box. For more information, see Assigning Help to Your Forms and Controls on page 263.

**Applies to:** Forms, all controls, except for frames.

**Setting dynamically:** No.

**Hyperlink Property**

The Hyperlink property determines whether the browser of the end user should treat a text control as a hyperlink. If the browser recognizes the text control as a hyperlink, then it displays a hand whenever end users move their cursors over the text control, and uses its own settings to determine what the text control will look like.

The text control must have an event handler assigned to the Click event. Otherwise, WebFOCUS Maintain ignores the setting for the Hyperlink property.

For example, in Internet Explorer 5, if you click Internet Options in the Tools menu, and then click Colors in the General tab, you can set the colors for visited and unvisited links and also turn on the Hover feature. Text controls with the Hyperlink property set to 1 - Yes will then use these settings.

**Applies to:** Text.

**Setting dynamically:**

Winform Set form.text.HYPERLINK TO {0 | 1};

**Property Settings:**

0 - No

   Turns off the Hyperlink property (the default).

1 - Yes

   Turns on the Hyperlink property.

**IBIValidation Property**

The IBIValidation property enables you to automatically validate data formats for edit boxes and multi-edit boxes at run time, as well as designate a field as required.
Applies to: Edit boxes and multi-edit boxes.

Setting dynamically:

Winform Set form.control.IBIValidation TO {0|1|2};

Property Settings:

0 – No

Disables the IBIValidation property. This is the default value.

1 – Yes

Enables the IBIValidation property. This will automatically validate the format of the data entered against the format of the field bound to the control.

If the edit box is bound to a field, a message will pop up at run time if the user enters data with a format that does not match the field (for example, alphanumeric data for a numeric field). The message appears when you submit the form.

2 – Required

Requires you to provide a value when this setting is applied. If you do not supply a value, a message appears when you submit the form indicating that data must be entered. After you enter a value, the format is validated.

Setting validation on and off dynamically using JavaScript or VBScript:

document.Formname.Controlname.ibivalidation={true|false}

Setting ‘Required’ dynamically using JavaScript or VBScript:

document.formname.controlname.MISSING={"ON"|"OFF"};

where:

OFF

Means a value is required.

ON

Means a value can be missing.

To set the Required option dynamically using JavaScript or VBScript, you must first enable basic validation, either from within the MDE before run time, or by using the script syntax noted above.
Image Property

The Image property determines the name of an image resource to appear on your form. To set the value for this property, use the Image Source dialog box. For more information, see Using Images on page 223.

Applies to: Images.

Setting dynamically: Yes. You can set its value to a Maintain variable.

ImageDown Property

The ImageDown property determines the image that appears when an end user clicks on an image. To set the value for this property, use the Image Source dialog box. For more information, see Using Images on page 223.

Applies to: Images.

Setting dynamically: Yes. You can set its value to a Maintain variable.

ImageOver Property

The ImageOver property determines the image that appears when end users move their mouse cursors over an image. To set the value for this property, you use the Image Source dialog box. For more information, see Using Images on page 223.

Applies to: Images.

Setting dynamically: Yes. You can set its value to a Maintain variable.

ItemBorder Property

The ItemBorder property determines whether a menu item in a menu has a border.

You can also set this property using a cascading style sheet. For more information, see Using Cascading Style Sheets on page 113.

Applies to: Menus.

Setting dynamically: No.

Layer Property

The Layer property designates which layer of the form a control is on. The default value is the Default layer. For more information, see Layering Controls on page 103.

Note: This is a form development property only. You cannot use it at run time.
**Left Property**

For controls, the Left property determines where the left side of a control is on the form, measured in pixels. Together with the Bottom, Top, and Right properties, it determines the size of your control and where it is on the form.

You can also set this property using a cascading style sheet. For more information, see *Using Cascading Style Sheets* on page 113.

**Applies to:** All controls.

**Setting dynamically:**

Winform Set control.LEFT TO n;

**ListItems Property**

The ListItems property determines the items an end user sees in a group of radio buttons, a combo box, or a list box. Set this value using the List Source dialog box. For more information, see *List Source Dialog Box* on page 176.

**Applies to:** Radio buttons, combo boxes, list boxes.

**Setting dynamically:** No.

**Map Property**

The Map property enables you to define individual parts of an image to be clickable. Set this information using the Image Map dialog box. For more information, see *Using Image Maps* on page 229.

**Applies to:** Images.

**Setting dynamically:** No.

**MultiSelection Property**

The MultiSelection property specifies whether end users can select multiple items in a list box.

**Applies to:** List boxes.

**Setting dynamically:**

Winform Set form.listbox.MULTISELECTION TO {0|1};
**Property Settings:**

\[ 0 \rightarrow No \]

Disables multi-selection feature (the default).

\[ 1 \rightarrow Yes \]

Enables the multi-selection feature.

**OnLoad Property**

The OnLoad property specifies a JavaScript to be run when the form is displayed.

**Applies to:** Forms.

**Setting dynamically:** No.

**Property Settings:**

Enter the name of the JavaScript file with the extension. For example, TestScript.js. The JavaScript file must be included in the project.

The file should not be embedded or linked. The javascript file will be available at local run time. For remote deployment it must be assigned to a web server.

The referenced JavaScript file must begin with:

```
function formOnLoad()
```

**Orientation Property**

The Orientation property determines whether a menu bar is displayed horizontally or vertically.

**Applies to:** Menus.

**Setting dynamically:**

```
Wiform Set form.menu.ORIENTATION To {0|1}
```

**Property Settings:**

\[ Horizontal \ (1) \]

Displays the menu bar horizontally (the default).

\[ Vertical \ (0) \]

Displays the menu bar vertically.
Overflow Property

The Overflow property determines how much of an HTML Object or HTML Table appears to an end user. By default, the entire contents of the HTML Object or HTML Table appear on the form. However, if you want to ensure that the HTML Object or HTML Table will never be larger than a certain size, then change the Overflow property to Clip or Scroll.

For example, you would do this if there was information underneath or directly beside the HTML Object or HTML Table and you wanted to ensure that the HTML Object or HTML Table would not overlap the information.

**Note:** In an HTML Table, the headers will also scroll.

**Applies to:** HTML Objects, HTML Tables.

**Setting dynamically:**

```
Winform Set form.HTMLcontrol.OVERFLOW TO n;
```

**Property Settings:**

- **0 - Visible**
  
  Displays all data in an HTML Object or HTML Table (the default). The HTML Table will be as wide and as long as necessary in order to display everything.

- **1 - Clip**
  
  Restricts the data to the length of the control as you drew it. Any rows that don’t fit are not displayed.

- **2 - Scroll**
  
  Restricts the visible data to what fits into the control as you drew it, but places scroll bars on the control (if necessary) so that the end user can view all of the data.

Password Property

The Password property determines what appears when an end user types information into an edit box. If the Password property is on, then the edit box displays asterisks (*), thus preventing observers from seeing what is being entered into the box.

**Applies to:** Edit boxes.

**Setting dynamically:**

```
Winform Set form.editbox.PASSWORD TO {0|1};
```
**Property Settings:**

0 - No

Displays the contents of the edit box (the default).

1 - Yes

Displays asterisks instead of the contents of the edit box.

**PenWidth Property**

The PenWidth property determines the width of a line, measured in pixels.

You can also set this property using a cascading style sheet. For more information, see *Using Cascading Style Sheets* on page 113.

**Applies to:** Lines.

**Setting dynamically:**

Winform Set form.line.PENWIDTH TO n;

**ReadOnly Property**

The ReadOnly property determines whether an end user can change the value being displayed in an edit box or multi-edit box.

**Applies to:** Edit boxes, multi-edit boxes.

**Setting dynamically:**

Winform Set form.control.READONLY TO {0|1};

**Property Settings:**

0 - No

Disables the read-only feature (the default).

1 - Yes

Enables the read-only feature.

**Right Property**

The Right property determines where the right side of a control is on the form. Together with the Top, Bottom, and Left properties, it determines the size of your control and where it is on the form.
 Rows Property

You can also set this property using a cascading style sheet. For more information, see Using Cascading Style Sheets on page 113.

**Applies to:** All controls.

**Setting dynamically:**

Winform Set form.control.RIGHT TO n;

Rows Property

The Rows property indicates the number of buttons in a column for a radio button. Together with the Columns property, this property determines the layout of your radio buttons.

The default value for this property is -1, meaning that WebFOCUS Maintain will use as many rows as it can fit in the radio button box.

For more information, see Determining the Layout of Your Radio Buttons on page 250.

**Applies to:** Radio buttons.

**Setting dynamically:**

Winform Set form.radiobutton.ROWS TO n;

Scrolling Property

The Scrolling property determines whether you can scroll in a frame.

**Applies to:** Frames.

**Setting dynamically:**

Winform Set form.frame.SCROLLING TO n;

**Property Settings:**

0 – Auto

   Enables scroll bars, if necessary (the default).

1 – No

   Disables scrolling.

2 – Yes

   Enables scroll bars.
**SelectedItem/SelectedItems Property**

The SelectedItem and SelectedItems properties set the value (what the end user selected) from a radio button control, combo box control, or list box control to a Maintain variable or data source stack column. Use the Binding the Selection Result dialog box to set this value. For more information, see *Binding the Selection Result Dialog Box* on page 179.

**Applies to:** Radio buttons, combo boxes, list boxes.

**Setting dynamically:** No.

**Source Property**

The Source property determines the contents of a frame. Selecting this property opens the URL Link dialog box. For more information, see *URL Link Dialog Box* on page 192.

**Applies to:** Frames.

**Setting dynamically:** No.

**Stretched Property**

The Stretched property determines whether you can adjust the size of an image. If you leave this property set to No, the image will be placed on the form in the same size as the original image file.

You can also set this property using a cascading style sheet. For more information, see *Using Cascading Style Sheets* on page 113.

**Applies to:** Images.

**Setting dynamically:**

Winform Set form.image.STRETCHED TO {0|1};

**Property settings:**

\[0\] - No

Does not allow you to resize the image (the default).

\[1\] - Yes

Enables you to resize the image.

**Tabstop Property**

The Tabstop property specifies whether an end user can tab to a control. When an end user tabs to a control, it has focus.
You can use the Tab Order dialog box to determine the tab order of controls on a form. For more information, see Assigning Tab Order to Controls on page 264.

**Applies to:** ActiveX controls, buttons, check boxes, combo boxes, edit boxes, list boxes, multi-edit boxes, radio buttons.

**Setting dynamically:**

Winform Set form.control.TABSTOP TO {0|1};

**Property Settings:**

0 - **No**

Disables the end user from tabbing to the control (the control is removed from tab order).

1 - **Yes**

Enables the end user to tab to the control (the default).

---

**Text Property**

The Text property determines what text appears in a control.

**Applies to:** Buttons, check boxes, edit boxes, group boxes, multi-edit boxes, text.

**Setting dynamically for button, check box, text:**

Winform Set form.control.TEXT TO "text";

**Setting dynamically for edit box and multi-edit box:** Yes. You can set its value to a Maintain variable.

---

**TextOnLeft Property**

The TextOnLeft property determines whether the description for a check box appears to the left or the right of the check box.

**Applies to:** Check boxes.

**Setting dynamically:**

Winform Set form.checkbox.TEXTONLEFT TO {0|1};

**Property Settings:**

0 - **No**

Places the text on the right (default).
Title Property

The Title property determines the title that appears on your form’s title bar at run time. The default is Untitled.

**Applies to:** Forms.

**Setting dynamically:**

Winform Set form.TITLE TO "text";

ToolTipText Property

The ToolTipText property enables you to display pop-up information about a form or control when end users place their cursors over the form or control.

**Note:** Internet Explorer 4 and 5 do not support the ToolTipText property for combo boxes and list boxes. However, since the World Wide Web Consortium spec for DHTML does specify this property, we have left it in the product in case a future release of the browser does support it.

**Applies to:** Forms, all controls.

**Setting dynamically:**

Winform Set form.[control.]TOOLTIPTEXT TO "text";

Top Property

For controls, the Top property determines where the top of a control is on the form, measured in pixels. Together with the Bottom, Left, and Right properties, it determines the size of your control and where it is on the form.

You can also set this property using a cascading style sheet. For more information, see *Using Cascading Style Sheets* on page 113.

**Applies to:** All controls.

**Setting dynamically:**

Winform Set control.TOP TO n;
Visible Property

The Visible property specifies whether a control is visible or invisible to an end user.

**Note:** This property can conflict with tab order.

You can also set this property using a cascading style sheet. For more information, see *Using Cascading Style Sheets* on page 113.

**Applies to:** All controls.

**Setting dynamically:**

```
Winform Set form.control.VISIBLE TO {0|1};
```

**Property Settings:**

- **0 - No**
  - Hides control from the end user.
- **1 - Yes**
  - Makes controls visible to the end user (default).

**Setting dynamically using JavaScript or VBScript:**

```
document.Formname.Controlname.style.visible="{true|false}";
```

ZIndex Property

The ZIndex property allows you to designate which control will be displayed on top of another control when more than one is overlaid on a form.

**Applies to:** All controls except Read/Write Grid, Menu control, and ActiveX controls.

**Setting dynamically:**

```
Winform Set form.button.ZIndex TO n
```

**Property Settings:**

Type an integer for ZIndex in the property sheet for the control. The control with the highest value will be displayed in the foreground. By default, the first ZIndex assignment is 0. The value for each successive control placed on a form will be incremented by 1. To ensure a particular control remains on top while you continue to add to your form, you can assign it a very high value for ZIndex (for example, 999).

**Setting dynamically using JavaScript or VBScript:**

```
document.Formname.Controlname.style.zIndex=n
```
Note: In JavaScript, the property must be written zIndex, with only the letter I capitalized.
Executing Other Procedures

Separating your application code into more than one procedure (called modularizing or partitioning) can significantly speed up your development time and application performance.

The Maintain language contains two commands for executing procedures:

- The CALL command executes other Maintain procedures.
- The EXEC command executes WebFOCUS procedures.

These two commands work more or less in the same way. Each command has some options that enable you to do the following:

- Determine the WebFOCUS Server where the called procedure is located.
- Determine whether to keep or close the server session when the procedure finishes executing (the server session is a process on the WebFOCUS Server that executes the procedure).
- Pass parameters to and receive parameters from the called procedure.

In this chapter:

- The Advantages of Modularizing Source Code
- Using the CALL and EXEC Commands
- Executing a Procedure on Another Server: AT Server
- Keeping or Terminating the Server Session: KEEP/DROP
- Passing Parameters Between Maintain Procedures: FROM...INTO
- Passing Parameters Between Maintain and WebFOCUS Procedures: FROM...INTO
- Optimizing Performance: Data Continuity and Memory Management

The Advantages of Modularizing Source Code

Separating your application code into more than one procedure has many advantages:

- You can use multiple procedures to share common source code among many developers, thus speeding up development and maintenance.
For example, a generalized error message display procedure could be used by all WebFOCUS Maintain developers. After passing a message to the generalized procedure, the procedure handles message display. The developers do not have to worry about how to display the message and error messages always look consistent to end users.

- You can remove infrequently-executed source code from a procedure and move it into its own procedure. This reduces the size of the original procedure, simplifying its logic, making maintenance easier, and using less memory if the new procedure is not called.
- You can split your application into presentation logic and database logic, and then run each type of logic on a server optimized to support that type of logic.

Using the CALL and EXEC Commands

To use the CALL or EXEC command, you must open the calling procedure in the Procedure Editor and enter the command inside of a function (CASE/ENDCASE command). You can type the command directly or use the Language Wizard.

Syntax: How to Use the CALL and EXEC Commands

The syntax for CALL is:

```
CALL Maintain_procedure [AT server] [KEEP|DROP] [PATH {VAR|LIST}] [FROM var_list] [INTO var_list] [;]
```

The syntax for EXEC is:

```
EXEC WebFOCUS_procedure [AT server] [KEEP|DROP] [PATH {VAR|LIST}] [FROM var_list] [INTO var_list] [;]
```

CALL is for Maintain procedures; EXEC is for WebFOCUS procedures. The optional parameters for this command enable you to do the following:

**AT server**

Specifies the WebFOCUS Server where the procedure resides. For more information, see *Executing a Procedure on Another Server: AT Server* on page 300.

**KEEP|DROP**

Determines the behavior of the server when the procedure finishes executing. KEEP is the default value. For more information, see *Keeping or Terminating the Server Session: KEEP/DROP* on page 301.
**FROM**

Enables you to receive parameters from the called procedure. For more information, see *Passing Parameters Between Maintain Procedures: FROM...INTO* on page 301 and *Passing Parameters Between Maintain and WebFOCUS Procedures: FROM...INTO* on page 312.

**INTO**

Enables you to pass parameters to the called procedure. For more information, see *Passing Parameters Between Maintain Procedures: FROM...INTO* on page 301 and *Passing Parameters Between Maintain and WebFOCUS Procedures: FROM...INTO* on page 312.

### Optional PATH Keyword for CALL and EXEC Statements

When issuing a CALL or EXEC statement, the optional PATH keyword is used to specify additional locations (search paths) the system should use when searching for dependent resources (Master files, imported modules, and so on). The path location names are application names existing within the APPROOT directory structure or application names that have been introduced with the APP MAP command.

The search path value can be in the form of a Maintain variable or list of literal values enclosed in quotes, as follows:

```
CALL Procedure AT server PATH "AppDir1 AppDir2 AppDir3" ;
EXEC Procedure AT server PATH MyVariable ;
```

**Procedure:** How to Introduce a PATH Keyword Into a CALL or EXEC Statement

The Developer Studio deployment process can also introduce the PATH keyword into a CALL or EXEC statement. This is done by the deployment scenario editor, which provides a dialog box to control what values are used for the application path. To access this dialog box:

1. Open the deployment scenario editor.
2. Right-click the server folder under Available Servers.
3. Select Set Server Paths from the menu. The following dialog box appears.

![Set Running Paths For LOOPBACK](image)

Items included in the Running Paths list appear as PATH values for those procedures that are invoked with the CALL or EXEC statements on the selected WebFOCUS Environment.

**Executing a Procedure on Another Server: AT Server**

The optional AT server phrase in the CALL or EXEC command determines where the calling procedure should look for the called procedure.

In most cases, you do not have to supply this phrase. WebFOCUS Maintain does it for you. When you define the deployment scenario for your project, you decide on which WebFOCUS Servers each procedure will run. Then, when you deploy your project, WebFOCUS Maintain places the appropriate AT server phrase in your CALL and EXEC statements, so that the application will work properly.
This information is useful for two reasons:

- If you edit the procedures that WebFOCUS Maintain deployed to the WebFOCUS Server, you will see the AT server phrases in these procedures.
- If you want to CALL or EXEC a procedure in your application that is not part of your project, you must manually include the AT server phrase in your CALL or EXEC statement, so that the calling procedure knows where to find this procedure.

**Keeping or Terminating the Server Session: KEEP/DROP**

When you CALL or EXEC a procedure, the WebFOCUS Server on which the procedure resides opens a server session for that procedure. You can use the KEEP or DROP keyword in the CALL or EXEC command to determine the behavior of that server session.

**KEEP**

Leaves the server session active for reuse by subsequent calls. KEEP is the default value.

**DROP**

Terminates the server session.

Depending on your application, setting one or the other can improve performance. For more information, see *Persistence Management* on page 356.

**Note:** The KEEP keyword in the CALL or EXEC command is different from the GOTO END KEEP command, described in *Optimizing Performance: Data Continuity and Memory Management* on page 313.

**Passing Parameters Between Maintain Procedures: FROM...INTO**

When you CALL a Maintain procedure, you can pass parameters to the called procedure, and receive variables and stacks back from the called procedure using the FROM and INTO keywords in the CALL command. The FROM keyword identifies the parameters being passed, and the INTO keyword identifies the parameters being received.

You can pass and receive both stacks and variables to and from the called procedure. The called procedure will have corresponding FROM and INTO phrases in its MAINTAIN command (the command that begins all Maintain procedures).

When you specify input and output variables for a procedure, WebFOCUS Maintain adds a Parameters folder to the list of options under the procedure and lists the input and output parameters in it.
**Note:** All user variables (both stacks and simple, or scalar, variables) are global to a function or procedure, but not global to the project. In other words, to protect them from unintended changes in other parts of a project, you cannot directly refer to a variable outside of the procedure in which it is found (with the exception of the FocError transaction variable). However, you can access the data of a variable in other procedures, simply by passing it as a parameter from one procedure to another.

### Specifying FROM and INTO Parameters in the Calling Procedure

You can specify FROM and INTO parameters in the calling procedure in two ways:

- Open the calling procedure in the Procedure Editor, and type the FROM and INTO phrases into the CALL command statement. For more information, see *How to Call a Maintain Procedure With FROM and INTO Parameters* on page 302.

- If you have already defined the FROM and INTO parameters in the called procedure, use the Language Wizard to generate the CALL statement with the appropriate FROM and INTO phrases. For more information, see *Using the Language Wizard to Specify Parameters When Calling a Maintain Procedure* on page 305.

### Specifying FROM and INTO Parameters in the Called Procedure

You can specify FROM and INTO parameters in the called procedure in two ways:

- Open the called procedure in the Procedure Editor and type the FROM and INTO phrases after the MAINTAIN statement (see *How to Call a Maintain Procedure With FROM and INTO Parameters* on page 302).

- If you have already defined the parameters (either as stacks or variables), use the Procedure Parameters dialog box. For more information, see *Specifying FROM and INTO Parameters for a Called Procedure Using the Procedure Parameters Dialog Box* on page 304.

### Syntax: How to Call a Maintain Procedure With FROM and INTO Parameters

The syntax for the calling procedure is:

```plaintext
CALL Maintain_procedure [AT server] [KEEP|DROP] [FROM var_list1] [INTO var_list2] ;
```

The syntax for the corresponding called procedure is:

```plaintext
MAINTAIN [FILE[S] filelist] [FROM var_list1] [INTO var_list2]
```
The following rules apply:

- Some parameter attributes must match in the CALL and MAINTAIN commands:
  - **Number.** The number of parameters in the two procedures must be identical.
  - **Sequence.** The order in which you name stacks and simple variables must be identical in the CALL and corresponding MAINTAIN commands.
  - **Data type.** Stack columns and simple variables must have the same data type (for example, integer) in both procedures.
  - **Stack column names.** The names of stack columns must match. If a column has different names in the calling and called procedures, it is not passed.

- Other attributes need not match:
  - **Stack and scalar variable names.** The names of stacks and simple variables specified in the two commands do not need to match.
  - **Other data attributes.** All other data attributes, such as length and precision, do not need to match.
  - **Simple variables.** If you pass an individual stack cell, you must receive it as a simple variable, not as a stack cell.
  - You cannot pass variables or data source stacks using the AO format or Stack of format.

**Defining FROM and INTO Parameters in Calling and Called Maintain Procedures**

After you pass a variable to a called Maintain procedure, you must define it in that procedure. The definition depends upon the type of variable:

- **User-defined columns and fields.** You must redefine each user-defined variable using a DECLARE or COMPUTE command. You must specify only the format of the variable, not its value. For example, the following DECLARE command redefines the Counter field and the FullName column:

  DECLARE Counter/A20;
  EmpStack.FullName/A15;

- **Data source and virtual stack columns.** You can define the data source of a column stack and virtual columns in one of two ways: implicitly, by referring to the stack columns in a data source command, or explicitly, by referring to them using the INFER command. For example:

  INFER Emp_ID Pay_Date INTO EmpStack;
The INFER command declares data source fields and the stack with which they are associated. You can specify one field for each segment you want in the stack, or simply one field each from the anchor and target segments of a path you want in the stack.

While INFER re-establishes the definition of a stack, it does not retrieve any records from the data source.

After a variable is defined in the called procedure, its data becomes available. If you refer to stack cells that were not assigned values in the calling procedure, they are assigned default values (such as spaces or zeros) in the called procedure, and a message warns you that they have not been explicitly assigned any values.

When the called procedure returns control back to the calling procedure, the values of stacks and simple variables specified as output arguments are passed back to the calling procedure. The values of stacks and simple variables specified only as input arguments are not passed back.

**Specifying FROM and INTO Parameters for a Called Procedure Using the Procedure Parameters Dialog Box**

When you right-click a procedure and click Add Parameters in the shortcut menu, WebFOCUS Maintain opens the Procedure Parameters dialog box, shown in the following image. Use this dialog box to specify variables and data source stacks whose values the procedure is receiving from and passing to a calling procedure.
Procedure: How to Specify Parameters for a Called Procedure Using the Procedure Parameters Dialog Box

1. Define the variables and data source stacks whose values will be passed to and from the procedure.
2. Right-click the procedure, and in the shortcut menu, click Add Parameters.
3. Select the variables and stacks that will be passed to the procedure from the calling procedure.
4. Click the right arrow button to move them into the Input parameters list.
5. If necessary, reorder the parameters. They must be in the same order here as they are in the CALL statement in the calling procedure. However, they do not need the same name.
6. Select the variables and stacks that will be passed back to the calling procedure.
7. Click the right arrow button to move them into the Output parameters list.
8. If necessary, reorder the parameters. They must be in the same order here as they are in the CALL statement in the calling procedure. However, they do not need the same name.
9. Click OK.

You will see a Parameters folder under your procedure with the input and output parameters.

Note: If you edit your procedure as text, your parameters are listed in the MAINTAIN line of code. The input parameters follow the FROM keyword. The output parameters follow the INTO keyword.

Using the Language Wizard to Specify Parameters When Calling a Maintain Procedure

If you have defined parameters for a called Maintain procedure, you can use the Language Wizard to call that Maintain procedure and specify corresponding parameters in the calling procedure.

Procedure: How to Use the Language Wizard to Specify Parameters When Calling a Maintain Procedure

1. Before you open the Language Wizard, you must define the variables and stacks you will be using as input and output parameters.
   You must also define the corresponding variables and stacks you will use.
2. Open your calling procedure in the Procedure Editor, and right-click in a function (between a CASE statement and an ENDCASE statement).
3. Click Language Wizard.
4. Select *Run another procedure* and click *Next*.

5. Select *Call* and click *Next*.

6. Select the procedure you want to call from the list (or click *Find other procedures* to look on your local drive or any WebFOCUS Server), and click *Next*.

7. Double-click on each parameter listed in the window and specify a corresponding variable or stack from this procedure, using the Parameter Value dialog box.

   Input parameters (which you pass to the called procedure) are represented by the following icon:

   ![Input Parameter Icon]

   Output parameters (the values the called procedure will return) are represented by the following icon:

   ![Output Parameter Icon]

8. Click *Next*.

9. Specify whether to keep or drop the server connection when the called procedure finishes executing, and click *Finish*.

**Example:** Passing Parameters Between Two Procedures

This example demonstrates how to pass a stack from a called procedure to a calling procedure, and the steps you must go through for both procedures to work properly.

Suppose you are designing a project that enables clerks at a video store to sell or rent videos to customers. The clerks access a WebFOCUS Server on a UNIX machine through the web, but the actual project data resides on an MVS machine. The project may have procedures running on both the UNIX machine and the MVS machine; the machines must call each other and pass data back and forth (this is a common occurrence in WebFOCUS Maintain).

Suppose that the project (called VideoApp) must display customer information to the clerk.

1. On the UNIX machine, the WebFOCUS server will run a procedure (called ShowCust) that calls a procedure (called GetCust) on the MVS machine.

2. GetCust extracts data from the Videotrk data source and places it in a data source stack called CustInfo.

3. GetCust passes CustInfo back to ShowCust.
ShowCust then displays a form with the information in CustInfo, as shown in the following image.

Here is the Maintain code for ShowCust:

```
MAINTAIN FILE Videotrk
$$Declarations
Case Top
Infer Videotrk.CUST.CUSTID into CustInfo;
Call GetCust Into CustInfo;
EndCase
END
```

Here is the Maintain code for GetCust:

```
MAINTAIN FILE Videotrk INTO CustInfo
$$Declarations
Case Top
For all next Videotrk.CUST.CUSTID into CustInfo;
EndCase
END
```

**Before you begin...**

1. Create a project named VideoApp, with the data source Videotrk (one of the sample data sources distributed with WebFOCUS).
2. Create two Maintain procedures: GetCust and ShowCust.
Specify that GetCust and ShowCust should use the Videotrk data source:

1. Right-click GetCust.
2. In the shortcut menu, click Use data sources.
3. In the Use these Data Sources in Procedure GetCust dialog box, shown in the image below, select Videotrk in the Available data sources list.

4. Click the right arrow button  ▶️  

5. Click OK.
6. Repeat the process for ShowCust.

Create the data source stack CustInfo in GetCust and load the data into it:

1. Double-click GetCust.
2. Between the Case Top and EndCase keywords, type the following Maintain language code:
   
   For all next CUST.CUSTID into CustInfo ;

   Tip: You can also create this code using the Language Wizard. Place your insertion point between the Case Top and EndCase keywords, right-click in the Procedure Editor window, and in the shortcut menu, click Language Wizard.

3. Close GetCust.

Note: You created the stack CustInfo implicitly when you loaded data into the stack.
**Make CustInfo an output parameter of GetCust:**

1. Right-click GetCust.
2. In the shortcut menu, click Add Parameters.
3. In the Procedure Parameters dialog box, shown in the image below, select CustInfo in the Available variables and stacks list.
4. Click the right arrow button to copy CustInfo into the Output parameters list.
5. Click OK.
6. WebFOCUS Maintain creates a Parameters folder under GetCust and adds CustInfo there. The red arrow pointing down indicates that CustInfo is an output parameter.

**Define CustInfo in ShowCust:**

Since CustInfo is a data source stack in GetCust, ShowCust has no information on the structure of CustInfo. Therefore, you must define a stack to receive its value. However, since you are not loading data into the stack here (rather, you are receiving the data from GetCust), all you have to do is define the columns.

1. Right-click ShowCust.
2. In the shortcut menu, click New.
3. In the submenu, click Data source stack.
4. In the Stack Editor dialog box, type CustInfo in the Stack Name box.

**Note:** In this example, you are giving the stack being passed between the two procedures the same name: CustInfo. However, you do not have to give the stacks the same name as long as they have compatible types.

5. Expand the fannames data source in the Available fields list.

6. Expand the CUST segment.

7. Select the CUSTID field.

8. Click the right arrow button. WebFOCUS Maintain copies all of the fields in the CUST segment into your stack definition along with CUSTID, as shown in the following image.

![Stack Editor dialog box](image)

9. Click OK.
**Call GetCust from ShowCust:**

1. Double-click *ShowCust*.

2. In the Procedure Editor, type the following Maintain language code in front of the keyword `EndCase`:

   `Call GetCust into CustInfo`

**Note:** You can also create this code using the Language Wizard. Place your insertion point in front of the `EndCase` keyword, right-click in the Procedure Editor window, and in the shortcut menu, click *Language Wizard*.

The data from the Videotrk data source is now available to you in ShowCust to display for the end user.

**Accessing Data Sources in the Called Procedure**

If a called Maintain procedure accesses a data source, whether retrieving or writing records, you must specify the data source in the `MAINTAIN` command. This is done the same way as a stand-alone procedure. For example, the procedure below specifies the Employee and EducFile data sources:

```
MAINTAIN FILES Employee AND EducFile FROM StuStk INTO CoursStk
```

**Data Source Position in Child Procedures**

Each Maintain procedure tracks its own position in the data source. When you first call a procedure, Maintain positions you at the beginning of each segment in each data source accessed within that procedure. After navigating through a data source, you can reposition to the beginning of a segment by issuing the `REPOSITION` command. Each data source position of a procedure is independent of the positions established in other procedures.

When a called procedure returns control to its calling procedure, it clears its data source positions by default. You can specify that it retain its positions for future calls by using the `KEEP` option, as described in *Optimizing Performance: Data Continuity and Memory Management* on page 313.
Passing Parameters Between Maintain and WebFOCUS Procedures: FROM…INTO

When you EXEC a WebFOCUS procedure, you can pass variables (not stacks) to the called procedure, and receive stacks back from the called procedure using the FROM and INTO keywords in the EXEC command. The FROM keyword identifies the variables being passed, and the INTO keyword identifies the stacks being received.

For more information on WebFOCUS procedures, see Using WebFOCUS Procedures in Your Application on page 315.

**Note:** You cannot use the Language Wizard to specify the FROM and INTO phrases; you must type them in by hand.

**Reference:** Usage Notes for Input and Output Parameters in the EXEC Command

When you are passing input variables to and receiving output from a WebFOCUS procedure, the following rules apply:

- The WebFOCUS procedure receives the input variables as numbered amper variables.
- You can pass any scalar variable except for those defined as variable-length character variables (that is, except for those defined as A0 or TX).
- You cannot pass stacks to the target procedure.
- The length of a single input parameter cannot exceed 32,000 characters. The total length of all specified input parameters cannot exceed 32,000 characters.
- The output parameters must be stacks.
- If the WebFOCUS procedure is a report request, it must have the statement ON TABLE PCHOLD or ON TABLE PCHOLD AS HTMTABLE in it.
- If there are multiple answer sets, define multiple stacks as output parameters.
- You can receive data in any stack, except for those defined using Stack of.
- If the number of stacks specified is fewer than the number returned, the extra returned answer sets are ignored.
Optimizing Performance: Data Continuity and Memory Management

By default, when you terminate a called Maintain procedure, Maintain clears its data from memory to save space. You can optimize your application’s performance by specifying, each time you terminate a called procedure, how you want Maintain to handle the data of the procedure. You have two options, based on how often you call a given procedure over the course of an application.

If you call the procedure frequently, use the KEEP option to make the procedure run faster by retaining its data between calls.

This option provides data continuity. The data carries over from the end of one invocation to the beginning of the next. The next time you call the procedure, its variables and data source position pointers start out with the same values that they held when the procedure was last terminated. You can use these values or reinitialize them using the DECLARE (or COMPUTE) and REPOSITION commands.

Variables passed by the parent procedure are not affected by data continuity, since the child procedure receives them directly from the parent procedure at the beginning of each call.

**KEEP affects transaction integrity in the following way.** The KEEP option does not issue an implied COMMIT command at the end of a child procedure. When a child procedure with an open logical transaction returns to its parent procedure and specifies KEEP, the transaction continues into the parent.

If you rarely call the procedure, use the RESET option to reduce memory consumption by freeing the data at the end of each call.

This option does not provide data continuity. All of the variables and data source position pointers of the procedure are automatically initialized at the beginning of each procedure.

**RESET affects transaction integrity in the following way.** The RESET option issues an implied COMMIT command at the end of a child procedure. When a child procedure with an open logical transaction returns to its parent procedure using RESET, the transaction is closed at the end of the child procedure.

For more information about transactions spanning procedures, see *Ensuring Transaction Integrity* on page 359.

You can specify how a procedure will handle its data in memory by terminating it with the GOTO END command qualified with the appropriate memory-management phrase. The syntax is

```
GOTO END [KEEP | RESET];
```
where:

**KEEP**
Terminates the procedure, but keeps its data, the values of its variables and data source position pointers in memory. It remains in memory through the next invocation, or (if it is not called again) until the application terminates. The procedure does not issue an implied COMMIT command to close an open logical transaction.

**RESET**
Terminates the procedure, clears its data from memory, and issues an implied COMMIT command to close an open logical transaction. RESET is the default value.

You can use both options in the same procedure. For example, when you are ready to end a child procedure, you can evaluate what logic the procedure has to perform when it is next called and then branch accordingly either to keep data in memory, saving time and providing data continuity, or else to clear data from memory to conserve space.

**Note:** If you call a procedure with the DROP option, WebFOCUS Maintain will clear the data from the called procedure from memory, no matter what option you choose in the called procedure.
Using WebFOCUS Procedures in Your Application

One of the possible components of a WebFOCUS Maintain project is a WebFOCUS procedure. A WebFOCUS procedure contains code written in the WebFOCUS language. Using WebFOCUS, you can create procedures that execute reports, reset WebFOCUS server settings, and call other types of proprietary procedures. Integrating WebFOCUS procedures can greatly increase the scope of your application.

In this chapter:

- What Can WebFOCUS Procedures Do?
- Incorporating WebFOCUS Procedures Into Your Project
- Executing WebFOCUS Procedures From Maintain Procedures
- Using WebFOCUS Report Output in Maintain
- Executing Maintain Procedures From WebFOCUS Report Procedures
- Using Variable Binding From WebFOCUS Reports

What Can WebFOCUS Procedures Do?

WebFOCUS procedures can include the following:

- Reports.
- Dialogue Manager calls.
- API function calls.
- Calls to other non-WebFOCUS procedures:
  - A compiled program, written in a language such as C or COBOL, which can be called on a WebFOCUS Server.
  - A transaction running under the control of a transaction processing monitor such as CICS or IMS/TM.
  - An executable file of commands written in a proprietary RDBMS language.
You can execute these other kinds of logic by issuing API function calls, such as EDARPC, CALLPGM, CALLORA, CALLSYB, and CALLIMS, in your WebFOCUS procedure.

For more information about WebFOCUS procedures, see the following manuals:

- Creating Reporting Applications With Developer Studio
- Creating Reports With WebFOCUS Language

### Incorporating WebFOCUS Procedures Into Your Project

The Procedures folder is one of four default folders that appear under a new WebFOCUS Maintain project. This is where any WebFOCUS procedures that you place in your project will go.

- To create a WebFOCUS procedure, see *How to Create a Project Component* on page 29.
- To add an existing WebFOCUS procedure to your project, see *How to Add an Existing File to Your Project* on page 31.

**Procedure:** How to Edit the Source Code of a WebFOCUS Procedure

To edit the source code of a WebFOCUS procedure:

1. Right-click the procedure.
2. In the shortcut menu, click *Open*.

WebFOCUS Maintain opens the WebFOCUS Procedure Editor, where you can make changes to your procedure.

However, it is recommended that you return to WebFOCUS Developer Studio, which contains a full suite of tools for editing your WebFOCUS procedure.

### Executing WebFOCUS Procedures From Maintain Procedures

You can execute a WebFOCUS procedure from a Maintain procedure using the EXEC command. For more information, see *Executing Other Procedures* on page 297.

**Note:** If you are using a FOCUS Database Server and will be accessing Maintain procedures and WebFOCUS report procedures on the same WebFOCUS Server, you must access them in a particular way to ensure the integrity of your logical transactions. For more information, see *Ensuring Transaction Integrity* on page 359.
Using WebFOCUS Report Output in Maintain

One of the most common tasks WebFOCUS procedures can accomplish for your application is executing a report. Include report output in your application in one of the following ways:

- Execute the report and return the results to Maintain using a one-column stack, which you display using an HTML Object. This method preserves the report formatting. For more information, see How to Display the Output From a FOCUS Report on a Form on page 317.

- Execute the report and return the results to Maintain using a stack (the stack columns are named after the columns in the report). You can then use the output from the report in your Maintain procedure (for example, you can display it using an HTML table or grid). For more information, see How to Use the Output From a WebFOCUS Report on page 318.

- Execute the report using a web link. For more information, see How to Use the URL Wizard to Define an HTTP Web Link on page 147.

Note: If you are using a FOCUS Database Server and are accessing Maintain procedures and WebFOCUS report procedures on the same WebFOCUS Server, you must access them in a particular way to ensure the integrity of your logical transactions. For more information, see Ensuring Transaction Integrity on page 359.

For more information on WebFOCUS reports, see the Creating Reporting Applications With Developer Studio and Creating Reports With WebFOCUS Language manuals.

Procedure: How to Display the Output From a FOCUS Report on a Form

1. Ensure the report contains the following line:

   ```
   ON TABLE PCHOLD FORMAT HTMTABLE
   ```

2. Include the WebFOCUS report procedure in your project.

3. Create a stack with one computed column named HTML, of the type A250.

4. Execute the report procedure using the following syntax:

   ```
   EXEC procedure [KEEP|DROP] INTO stack;
   ```

   For more information on KEEP versus DROP, see Persistence Management on page 356.

5. Display the contents of the stack using an HTML Object.

For an example of this method, see WebFOCUS Maintain Advanced Tutorial in the Maintain Getting Started manual.
**Procedure: How to Use the Output From a WebFOCUS Report**

1. Ensure the WebFOCUS report procedure contains the following line:
   
   ```plaintext
   ON TABLE PCHOLD
   ```

2. Include the WebFOCUS report procedure in your project.

3. Create a stack with columns that match the names of the columns in the report.

   **Note:** If you do not know the names of the columns in your report, create a HOLD file, and then open the file description for the HOLD file to see the column names.

4. Execute the WebFOCUS report procedure using the following syntax:

   ```plaintext
   EXEC procedure [KEEP|DROP] INTO stack;
   ```

   For more information on KEEP versus DROP, see *Persistence Management* on page 356.

   **Note:** The default format of the WebFOCUS Server on which the report is executed must be HTML.

**Example: Using an HTML Table to Display the Results of a WebFOCUS Report**

Suppose you created the following report using WebFOCUS Developer Studio and included it in your project:

```plaintext
- * File FANREPT.FEX

TABLE FILE FANNAMES
ON TABLE SET PAGE-NUM OFF
ON TABLE SET PRINT ONLINE
PRINT FIRSTNAME AND COMPANY AND EMAIL BY LASTNAME
HEADING
"PAGE <TABPAGENO  
ON TABLE NOTOTAL
-* This command directs WebFOCUS to hold the report results in a stack.
ON TABLE PCHOLD
ON TABLE SET STYLE *
UNITS=IN, PAGESIZE='Letter', LEFTMARGIN=0.250000, RIGHTMARGIN=0.250000,
TOPMARGIN=0.250000, BOTTOMMARGIN=0.250000, SQUEEZE=ON,
ORIENTATION=PORTRAIT, $
TYPE=REPORT, GRID=ON, FONT=TIMES NEW ROMAN, SIZE=10, COLOR=BLACK,
BACKCOLOR=NONE, STYLE=NORMAL, $
ENDSTYLE
END
```
This report displays the Firstname, Company, and Email fields from the Fannames data source, sorted by the Lastname field. Note that the syntax contains the ON TABLE PCHOLD line.

You can execute this report from a WebFOCUS Maintain procedure and save the results in a data source stack. The names of the columns in the data source stack must correspond to the columns in the report (that is, FIRSTNAME, COMPANY, EMAIL, and LASTNAME).

1. Create a Maintain procedure.
2. Use the Fannames data source in your procedure:
   a. Right-click the procedure and click Use data sources.
   b. In the Use these Data Sources in Procedure dialog box, click the Display all files in the project paths button.
   c. Select fannames and click the right arrow button.

   If fannames does not appear in the project path, see How to Add an Existing File to Your Project on page 31.

3. Create a new data source stack by right-clicking the Maintain procedure, clicking New, and then clicking Data Source stack.

4. In the Stack Editor, name the stack FanStack.

5. In the Available fields box, expand the Fannames data source, and then the CUSTOMER segment.
6. Using the right double arrow, move the SSN field to the *Stack columns* box, as shown in the following image.

![Stack Editor Image]

7. Click OK. You have defined the data source stack that receives the contents of the WebFOCUS report.

8. Open the Maintain procedure in the Procedure Editor.

9. After the line of Maintain code that defines FanStack (INFER), but before the Maintain code that displays Form1 (Winform Show), enter the following Maintain code:

```plaintext
EXEC FANREPT INTO FANSTACK;
```

You can type all of this code directly or you can generate the EXEC FANREPT part, either with the Language Wizard or by dragging FANREPT from the Project Explorer into the Procedure Editor. You must type INTO FANSTACK by hand.
10. Open Form1.

11. Select the *HTML Table* control on the Controls palette.

12. Draw a large rectangle on Form1.

WebFOCUS Maintain automatically opens the Control Columns dialog box. Ensure *FanStack* is selected in the list of data source stacks.

13. Copy *FIRSTNAME, LASTNAME, COMPANY, and EMAIL* to the Table columns box, as shown in the following image.

![Control Columns dialog box](image)

14. Click OK.
Now, when you run your application, WebFOCUS Maintain executes FANREPT and returns the results to FanStack. The HTML Table on Form1 displays the contents of FanStack, as shown in the following image.

![Image of HTML Table on Form1](image-url)

Any formatting information in the report is lost (if you wish, you can apply new formatting using the Table Column dialog box). To view formatting inherent in the report, use the ON TABLE HOLD HTMTABLE option, instead of ON TABLE PCHOLD and view the results in an HTML Object.

**Executing Maintain Procedures From WebFOCUS Report Procedures**

You can link reports developed in WebFOCUS Developer Studio to Maintain procedures using the drill down to Maintain Procedure feature.

**Procedure:** **How to Execute Maintain Procedures From WebFOCUS Report Procedures**

To execute Maintain procedures from WebFOCUS report procedures:

2. Select any component of the report, and click **Options** in the Properties menu.
3. Click the **Drill down** tab.
4. Select **Maintain procedure**.
5. Type the name of the procedure the report will call. This name must match the name of the Maintain procedure you are creating.
6. Add the parameters that will be passed to the Maintain procedure.
For more information on creating WebFOCUS reports, see the Creating Reporting Applications With Developer Studio manual.

**Using Variable Binding From WebFOCUS Reports**

When you execute a WebFOCUS report, you can pass parameters from the report to the Maintain procedure. For example, an end user can click on a row in a WebFOCUS report, which then runs a Maintain procedure enabling the end user to edit the data for that row.

The variable binding feature in WebFOCUS Maintain enables developers to write code that passes the parameters from the WebFOCUS report to the Maintain procedure quickly and easily, without even needing to know the Maintain language.

**Note:** In WebFOCUS Developer Studio, passed name/value pairs are called **parameters**. In Maintain, replaceable named values are called **variables**.

**Procedure: How to Use the Variable Binding Feature**

Before you can use variable binding in a Maintain procedure, you must create a WebFOCUS report that has a drill down to the Maintain procedure. For more information, see *How to Execute Maintain Procedures From WebFOCUS Report Procedures* on page 322.

After you create your report and specify the drill down parameters, do the following:

1. In the Maintain Development Environment, open the Maintain procedure in the Procedure Editor.
2. Place your cursor in the $$Declarations$$ section.
3. Right-click, and select **Import WebFOCUS Parameters**.
4. In the Open dialog box, select the WebFOCUS procedure from which you want to import parameters and click **OK**.
5. In the Import WebFOCUS Parameters into Maintain dialog box, assign new data types to the parameters you want to import using the Type Wizard and click **OK**. For more information, see *Using the Type Wizard* on page 78.

WebFOCUS Maintain places the following code at the beginning of your procedure:

```plaintext
Declare variable / A4 = IWC.GetAppCgiValue ( "variable" ) ;
```

**Reference:** Variable Binding Limitations

- If you import a parameter from a WebFOCUS report into an existing Maintain procedure, you must ensure that the parameter name matches the variable name that you are using elsewhere in the procedure.
For example, the procedure will not work if you use a variable named CustomerNo to determine which record is pulled from a database of customers, but you use CustomerID as the parameter name in the WebFOCUS report that launches the procedure.

You can, however, manually change the variable name in the Maintain procedure to match the parameter name.

- You cannot import parameters from a WebFOCUS report that is drilling down to a different Maintain procedure.

**Reference:** Importing WebFOCUS Parameters Into Maintain Dialog Box

When you import the parameters from a WebFOCUS report into a Maintain procedure, use the Import WebFOCUS Parameters Into Maintain dialog box, shown in the image below, to change the formats of the parameters.

This dialog box includes the following options:

**Name**

Contains the names of the parameters from the WebFOCUS report that are calling this Maintain procedure.
**Type**

Contains the proposed format for the parameter.

Opens the Type Wizard, where you can specify a new data type for the parameter. For more information, see *Using the Type Wizard* on page 78.
Using Variable Binding From WebFOCUS Reports
Most application development is modular: the developer creates complex systems comprised of smaller parts. In procedural development, these modules are procedures, and data is defined within each procedure. In object-oriented development, the modules are models of real-world objects (such as a customer or a shipping order), and both data and procedures are defined within each object. The object encapsulates the data and the procedures.

For example, if you are developing an order fulfillment system for a mail-order clothing business, the objects might include customers, orders, and stock items. The data of a customer object might include the customer ID code, phone number, and order history. The customer processes might include functions that add the customer to a new mailing list, update the customer information, and place an order for the customer.

Object-oriented development is a more efficient way of developing applications because it models the real-world objects with which your enterprise deals, and encourages you to reuse application logic in a variety of ways. You can use WebFOCUS Maintain to create applications using object-oriented development, procedural development, or a hybrid of these two methods, providing you with a flexible development path.

In this chapter:

- What Are Classes and Objects?
- Defining Classes
- Reusing Classes: Class Libraries
- Declaring Objects

What Are Classes and Objects?

Most applications need many objects of the same type. For example, if your business has 500 customers, you need one object to represent each customer. No one would want to design a customer object 500 times. Clearly, you need a template that defines all customer objects, so that you can design the template once, and use it often. For example, you would use the template each time you create a new customer object to represent a new customer.
An object template is called its class. Each object is an instance of a class. In other words, the class defines the type of object. In fact, when you create a class, the class becomes a new data type. Just as you can use a built-in data type, such as integer or alphanumeric, to define a simple variable, you can use a class data type to define an object.

Unlike a Master File, which is also a kind of template, a class defines both variables and functions. Just as a built-in data type defines the operations that you can perform on data of that type (for example, you can perform addition, subtraction, division, and multiplication on integers), a class defines the functions that you can perform on objects of that class (for example, you can invoke functions to update an address and to place an order for a customer object).

**Example: Comparing Classes and Built-in Data Types**

Just as you can use the alphanumeric built-in data type to define a customer ID code as an A8 variable

```
DECLARE CustID/A8;
```

you can use the RetailCustomer class to define a customer as an object:

```
DECLARE CustSmit8942/RetailCustomer;
```

**Class Properties: Member Variables and Member Functions**

You define a class by describing its properties. Classes have two kinds of properties:

- **Data**, in the form of the variables of a class. Because these variables exist only as members of the class, they are called member variables. In some object-oriented development environments these are also known as object attributes or instance variables.

  A class's member variables determine what the class is (as opposed to what it does). Each object of that class can have different values for its member variables.

- **Processes**, implemented as functions. Because these functions exist only as members of the class, they are called member functions. In some object-oriented development environments, these are also known as methods.

  The member function of a class defines its behavior. They determine what you can do to objects of that class, and in what ways you can manipulate the data.
**Example: Member Variables for a Customer Class**

An application for a mail-order clothing business has defined a customer class named Customer. The member variables of the class might include the customer's code, name, phone number, and most recent order number of the customer:

```
DESCRIBE Customer =
(IDcode/A6,
 LastName/A15,
 FirstName/A10,
 Phone/I10,
 LastOrder/A15);
.
.
ENDDESCRIBE
```

After declaring a new customer object for the customer, Frances Smith

```
DECLARE CustFrSmith/Customer;
```

you can assign a value to the IDcode member variable:

```
DECLARE CustFrSmith.IDcode = GetNewCustCode();
```

Each object can have different values for its member variables. For example, in this case, each customer will have a different ID code.

**Example: Member Functions for a Customer Class**

An application for a mail-order clothing business has defined a customer class named Customer. The class member functions might include a function that adds the customer to a new mailing list, a function that updates the customer contact information, and a function that places an order for the customer:

```
DESCRIBE Customer =
(IDcode/A6,
 Phone/I10,
 .
 .
 .
 LastOrder/A15);
 CASE AddToList TAKES Name/A25, Address/A50, IDcode/A6;
 .
 .
 .
ENDCASE
 CASE UpdateContact ...
```
After declaring a new customer object for the customer Frances Smith

```c
declare custfrsmith/customer;
```

you could add Frances Smith to the mailing list, using the AddToList member function:

```c
custfrsmith.addtolist();
```

Each object has the same member functions, and therefore, the same behavior. In this case, for example, each customer will be added to the mailing list using the function.

**Inheritance: Superclasses and Subclasses**

If you want to create a new class that is a special case of an existing class, you could derive it from the existing class. For example, in a human resources application, a class called Manager could be considered a special case of a more general class called Employee: all managers are employees, and possess all employee attributes, plus some additional attributes unique to managers. The Manager class is derived from the Employee class, so Manager is a subclass of Employee, and Employee is the superclass of Manager.

A subclass inherits all of its superclass properties. For example, it inherits all of the member variables and member functions of the superclass. When you define a subclass, you can choose to override some of the inherited member functions, meaning that you can recode them to suit the ways in which the subclass differs from the superclass. You can also add new member functions and member variables that are unique to the subclass.

**Defining Classes**

Before you can declare an object (an instance of a class), your procedure must have a class definition for that type of object. If the class:

- **Is already defined in a class library**, simply import the library into your procedure. Class libraries, which are implemented as import modules, are described in *Reusing Classes: Class Libraries* on page 340.

- **Is already defined in another procedure**, simply copy and paste the definition into a class library. You can then import the library into any procedure that needs it.

- **Is not yet defined anywhere**, you can define it in a class library or procedure using the Class Editor, or by coding the definition directly in the Procedure Editor, using the DESCRIBE command. If you define it in a class library, you can use the class definition in multiple procedures by simply importing the library into those procedures.
Procedure: How to Define a Class Using the Class Editor

This procedure describes how to define a new class. If you wish to define a new subclass, that is, a class that inherits properties from another class, see How to Define a Subclass Using the Class Editor on page 332.

1. Select the import module or procedure in which you want the class to reside.
2. Right-click the import module or procedure, click New in the shortcut menu, and click Class (Describe) in the submenu.
   or
   Click the New class button on the Application toolbar.
3. In the New Class dialog box, type a name for your class.
4. Click the Variables tab to specify the member variables of a class. The member variables of a class express its properties.
5. To add a variable, click the New button.
   The Member Variable dialog box opens. For instructions on how to create a variable, see How to Create a Variable in a Procedure on page 68.
   The name of each member variable must be unique within the class to which it belongs. It can be identical, however, to the names of member variables of other classes.
6. Repeat step 5 to create any additional variables.
7. Click the Functions tab to specify the member functions of a class. The member functions of a class define the actions that can be performed on the objects of a class.
8. To add a function, click the New button.
   The Member Function dialog box opens. For instructions on creating functions, see How to Create a Function on page 61.
   The name of each member function must be unique within the class to which it belongs. It can be identical, however, to the names of member functions of other classes.
9. Repeat step 8 to create any additional functions.
10. Optionally, click the Description tab and add a description to your class.
11. Click OK to confirm the class definition.
**Procedure: How to Define a Subclass Using the Class Editor**

To define a new class (a subclass) by inheriting properties (member functions and member variables) from another class (a superclass):

1. Select the import module or procedure in which you want the class to reside.
2. Right-click the import module or procedure, click New in the shortcut menu, and click Class (Describe) in the submenu.
   
   or
   
   Click the New class button on the Application toolbar.
3. In the New Class dialog box, type a name for your class.
4. Select the class whose properties the new class will inherit from the Inherits behavior from list. You can choose from all of the classes that are defined in this import module or procedure, and in any modules that have been imported into it. The selected class will be the superclass, and the new class will be the subclass.
5. Click the Variables tab to specify the member variables of a class. The member variables of a class express its properties. A subclass inherits all the member variables of its superclass, and you can add new ones.
6. To add a variable, click the New button.
   
   The Member Variable dialog box opens. For instructions on creating variables, see How to Create a Variable in a Procedure on page 68.

   **Note:** You cannot delete member variables inherited from the superclass.

7. Repeat step 6 to create any additional variables.
8. Click the Functions tab to specify the member functions of a class. The member functions of a class define the actions that can be performed on the objects of a class.

   Note that you cannot delete member functions inherited from the superclass. However, you can override an inherited member function it to edit or remove its source code.

9. To override an inherited function so that you can later edit its source code, select the function and then click the Override button.

10. To add a function, click the New button.

    The Member Function dialog box opens. For instructions on creating functions, see How to Create a Function on page 61.
The name of each member function must be unique within the class to which it belongs. It can be identical, however, to the names of member functions of other classes.

11. Repeat step 10 to create any additional functions.

12. Optionally, click the Description tab and add a description of the class. This description will be generated as a comment with the class's source code.

13. Click OK to confirm the class definition.

**Syntax:** How to Define a Class or Subclass Using the DESCRIBE Command

When you define a class using the Class Editor, it generates the definition in the procedure as a DESCRIBE command. If you wish to work directly with source code, you can create new class definitions and edit existing definitions directly in the Procedure Editor by using the following DESCRIBE syntax. You must issue the DESCRIBE command outside of a function, for example, at the beginning of the procedure prior to all functions.

```plaintext
DESCRIBE classname = ( [superclass +] memvar/type [, memvar/type] ...) [;]
[memfunction
[memfunction]...
ENDDESCRIBE]
```

where:

- **classname**
  - Is the name of the class that you are defining. The name is subject to the standard naming rules of the Maintain language. For more information, see the Maintain Language Reference manual.

- **superclass**
  - Is the name of the superclass from which you wish to derive this class. Include only if this definition is to define a subclass.

- **memvar**
  - Names one of the member variables of the class. The name is subject to the Maintain language standard naming rules. For more information, see the Maintain Language Reference manual.

- **type**
  - Is a data type (a built-in format or a class).
memfunction

Defines one of the class member functions. Member functions are defined the same way as other Maintain functions, using the CASE command. For more information, see the Maintain Language Reference manual.

;

Terminates the definition if the definition omits member functions. If it includes member functions, the semicolon is omitted and the ENDDESCRIPTION command is required.

ENDDESCRIBE

Ends the class definition if it includes member functions. If it omits member functions, the ENDDESCRIPTION command must also be omitted, and the definition must be terminated with a semicolon (;).

Procedure: How to Edit a Class Definition

To add a new member function or member variable:

1. Right-click the class in the Project Explorer, click New member, and then click Function or Variable.

2. In the New Function or New Variable dialog box, create your new function or variable. For instructions on creating functions, see How to Create a Function on page 61. For instructions on creating variables, see How to Create a Variable in a Procedure on page 68.

To edit one of the member functions or member variables of a class:

1. Right-click one of the class member functions or member variables in the Project Explorer.

2. In the shortcut menu, click Edit.

3. Make any necessary changes to the class’s definition in the Edit Variable or Member Function dialog boxes.

   For information about editing variables, see How to Edit a Variable on page 68. For information about editing functions, see How to Edit a Function on page 62. For general information about editing a class definition, see Defining Classes on page 330.

4. Click OK to confirm your changes.

Procedure: How to Edit a Class's Source Code

The Maintain Development Environment generates Maintain language code for classes. If you wish, you can edit this code directly in the Procedure Editor.

1. Right-click the class in the Project Explorer.

2. In the shortcut menu, click Go to definition.
3. Make any changes you wish to the code between DESCRIBE classname and ENDDESCRIPTION. For more information about the DESCRIBE command, see the Maintain Language Reference manual.

4. Close the Procedure Editor.

**Procedure: How to Rename a Class, Member Variable, or Member Function**

1. Right-click the class, member variable, or member function, and in the shortcut menu, click *Rename*.
   
   or
   
   Select the class, member variable, or member function, and press the F2 key.
   
   or
   
   Click the class, member variable, or member function twice.
   
2. Type the new name.
3. Press the Enter key to confirm the new name.

**Procedure: How to Delete a Class, Member Variable, or Member Function**

1. Select the class, member variable, or member function in the Project Explorer.
2. Right-click the class, member variable, or member function, and in the shortcut menu, click *Delete*.
   
   or
   
   Press the Delete key.
   
   or
   
   Click the Delete button on the General toolbar.
**Reference: New Class and Edit Class Dialog Boxes: General Tab**

The New Class and Edit Class dialog boxes enable you to create and edit classes. An example of the General tab is shown in the following image.

![New Class Dialog Box](image)

The General tab has the following options:

**Class name**

Type the name of your class here. For more information about names, see the *Maintain Language Reference* manual.

**Inherits behavior from**

If this class is based on another class, select that class from the list. The list comprises all of the classes that are defined in the open import module or procedure, and in any modules that have been imported into it.
Reference: New Class and Edit Class Dialog Boxes: Functions Tab

The New Class and Edit Class dialog boxes enable you to create and edit classes. An example of the Functions tab is shown in the following image.

The Functions tab includes the following options:

Member Functions

Lists the names of the member functions of a class.

Opens the Member Function dialog box, where you can define a new function.

Deletes a selected function from the list of functions, or if a function has been overridden, deletes the override.
Defining Classes

Moves a selected function up in the list of functions.

Moves a selected function down in the list of functions.

Overrides the selected inherited function. You would override a function inherited from a superclass if you wish to make changes to that function in this subclass.

**Reference:** New Class and Edit Class Dialog Boxes: Variables Tab

The New Class and Edit Class dialog boxes enables you to create and edit classes. An example of the Variables tab is shown in the following image.
The Variables tab includes the following options:

**Member Variables: (Name/Type)**

- Lists the class member variables, including those that it has inherited.

- Opens the Member Variable dialog box, where you can define a new variable.

- Deletes a selected variable from the list of variables. You cannot delete variables that are inherited from another class definition.

- Moves a selected variable up in the list of variables.

- Moves a selected variable down in the list of variables.
Reference: New Class and Edit Class Dialog Boxes: Description Tab

The New Class and Edit Class dialog boxes enables you to create and edit classes. An example of the Description tab is shown in the following image:

The Descriptions tab has the following elements:

**edit box**

You can document the class by typing a description of it in this edit box. Maintain will turn your description into a comment in the class definition source code in the procedure or module.

Reusing Classes: Class Libraries

You can define a class once, but use it in multiple Maintain procedures by storing its definition in a class library. Libraries are a very useful way of reusing source code, enabling you to develop applications more efficiently.

A class library is implemented as an import module (a kind of non-executable procedure) in which you can store class definitions (as well as Maintain functions). After you have created a module, you can import it into each Maintain procedure in which you want to use those classes. For information about creating, editing, and importing modules, see Using Import Modules on page 81.
After you have created the import module, simply create new class definitions in the module, or copy existing definitions into the module. You create and edit class definitions in a module in the same way that you create and edit them in a procedure, as described in Defining Classes on page 330.

You can nest modules to any depth. For example, if you have two import modules named ClasLib1 and ClasLib2, you can import ClasLib1 into ClasLib2.

**Note:** A library cannot contain an explicit Top function, and cannot refer to data sources. For example, class definitions in a library cannot contain data source commands (such as NEXT and INCLUDE) and cannot refer to data source stacks.

### Syntax: How to Import a Class Library Using the MODULE IMPORT Command

You can use the MODULE command to import libraries containing class definitions so that the current procedure can use those classes. Libraries can also contain other source code, such as function definitions. The syntax is

```plaintext
MODULE IMPORT (library_name [, library_name] ... );
```

where:

- `library_name`
  - Is the name of the Maintain procedure that you wish to import as a source code library.
  - Specify its file name without an extension. The file must reside in the path defined by the EDASYNR environment variable.

The MODULE command must immediately follow the MAINTAIN command.

### Declaring Objects

After a class definition exists, you can declare objects of that class. This is identical to declaring simple variables of a built-in data type. You can declare objects using the Class Editor, or by coding the declaration directly in the Procedure Editor.

### Procedure: How to Declare an Object Using the Variable Editor

**Prerequisite:** When declaring an object (that is, a class instance), the procedure in which you are declaring it must already include or import the class definition.

To declare an object using the Variable Editor:

1. Select the procedure in which you want the object to be declared.
2. Right-click the procedure, click New in the shortcut menu, and click Variable (Declare) in the submenu.

   or

   Click the New variable button on the Application toolbar.

3. In the New Variable dialog box, type a name for your object in the Name field.

4. Open the Type drop-down combo box. If the class of which this object will be an instance:
   - Is listed in the Type combo box, select it, and skip to step 9.
   - Is not listed in the Type combo box, and you know its name, enter the name in the Type combo box and skip to step 9.
   - Is not listed in the Type combo box, and you wish to select it from a list of all the procedure’s classes, click the ellipsis button to open the Type Wizard and continue with step 5. For more information, see Using the Type Wizard on page 78.

5. Select User-defined Class from the drop-down list.

6. From the drop-down list select Simple for a single object, or Stack of for a list of objects.

7. Select a class from the list.

   This adds the class to the list of data types in the Type combo box in the New Variable dialog box, making it available to you when you create additional objects in the future.

8. Click OK to return to the New Variable dialog box.

9. Optionally, click the Description tab and enter a description. This description will be generated as a comment with the object declaration’s source code.

10. Click OK to confirm the object declaration.

11. The Variable Editor created a global object declaration. If you wish to convert this to a local object declaration, while in the Procedure Editor simply cut the declaration from the $$Declarations section at the top of the procedure, and paste the declaration to the desired function.

    Note that local declarations must immediately follow the function’s CASE command, preceding all the other commands in the function.

**Syntax:** How to Declare an Object Using the DECLARE Command

You can declare a local or global object in the Procedure Editor using the DECLARE command. To make the declaration:
Local, code the DECLARE command in the function to which you want it to be local, following the function's CASE command, and preceding all the other commands in the function.

Global, code the DECLARE command outside of any function. It is recommended that you use the $$Declarations section at the top of the procedure to make the declaration easier for you to find.

You can also create global objects using the COMPUTE command. For information about the COMPUTE command, see the Maintain Language Reference manual.

To declare an object in the Procedure Editor using the DECLARE command, use this syntax

```maintain
DECLARE
[[]
objectname/class;
.
.
[].
[]]

where:

objectname
  Is the name of the object that you are creating. The name is subject to the standard naming rules of the Maintain language. For more information, see Specifying Names in Language Rules Reference in the Maintain Language Reference manual.

class
  Is the name of the class of which this object will be an instance.

[ ]

Groups a sequence of declarations into a single DECLARE command. The parentheses are required for groups of local declarations. Otherwise, they are optional.
Declaring Objects
You can start running a WebFOCUS Maintain application from the Maintain Development Environment. However, your end users will probably not have this tool on their computers, so you need another method for launching your WebFOCUS Maintain applications.

Start the WebFOCUS Maintain application by launching the starting object. This could be the initial Maintain procedure in the project, or it could be a launch form that executes the initial Maintain procedure.

**In this chapter:**

- Compiling WebFOCUS Maintain Procedures
- Executing Maintain Procedures From Outside the Maintain Development Environment
- Security and Running WebFOCUS Maintain Applications
- Customizing Key Functions and JavaScript for WebFOCUS Maintain Applications
- Closing WebFOCUS Maintain Applications

**Compiling WebFOCUS Maintain Procedures**

To speed up execution, we recommend compiling your Maintain procedures for a production environment. You can compile a procedure in one of two ways:

- At deployment time, you can specify that you want your Maintain procedures compiled by selecting *Compile* in the deployment scenario Properties dialog box.

- Using the Developer Studio Command Console, you can execute the following code:

  ```
  MNTCON COMPILE [appname/]procname
  ```

  where:

  - **appname**
    - Is the directory name on the WebFOCUS Server where the Maintain procedure is located (optional).

  - **procname**
    - Is the name of a Maintain procedure (with either an .mnt or .fex extension, or MAINTAIN or FOCCOMP file type or ddname).
**Note:** Issue an APP HOLD *application_name* command before the MNTCON COMPILE command to control where the FOCCOMP file will be stored.

For more information on the WebFOCUS Developer Studio Command Console, see the *Developing Reporting Applications* manual.

A compiled procedure has the extension .fcm or FOCCOMP file type or ddname.

### Executing Maintain Procedures From Outside the Maintain Development Environment

You can execute a WebFOCUS Maintain procedure from any of the following:

- A launch form
- A URL
- A WebFOCUS procedure
- The WebFOCUS Developer Studio Command Console
- A back-end server edastart –t session

### Syntax: How to Execute a WebFOCUS Maintain Procedure

To execute a WebFOCUS Maintain procedure that has not been compiled (with either an .mnt or .fex extension, or MAINTAIN or FOCEXEC file type or ddname), use the following syntax

```
MNTCON EX [appname/]procname [–v "parm1", ... "parmN"]
```

To execute a WebFOCUS Maintain procedure that has been compiled (with either a .fcm extension or FOCCOMP file type or ddname), use the following syntax

```
MNTCON RUN [appname/]procname [–v "parm1", ... "parmN"]
```

where:

- **appname**
  - Is the directory name on the WebFOCUS Server where the Maintain procedure is located (optional).

- **procname**
  - Is the name of a Maintain procedure.

- **–v**
  - Is the flag that indicates parameters will be passed to the Maintain procedure (optional).
parm1 ... parmN

Can be either positional parameters or parm="value" keyword parameters. Parameter types can be mixed within the same MNTCON EX or MNTCON RUN command line. The maximum number of parameters you can pass is 128. You should separate all parameters using commas. You should enclose values containing spaces or commas in single or double quotes. Use with Sys_mgr functions (Sys_Mgr.Get_NameParm, Sys_Mgr.Get_InputParams_Count and Sys_Mgr.Get_PositionParm) to retrieve the values. For more information, see MNTCON EX, MNTCON RUN, and SYS_MGR in the Command Reference in the Maintain Language Reference manual.

You can use the Developer Studio HTML Layout Tool to easily create your own launch form for your WebFOCUS Maintain applications by selecting the desired .mnt or .fcm file from the list of external procedures in the Create a Hyperlink dialog. For more information, see the Creating Reporting Applications with Developer Studio manual.

**Example:** Invoking an Uncompiled Maintain Procedure

MNTCON EX MYAPP

**Example:** Passing Parameters to a Compiled Maintain Procedure

MNTCON RUN start1 -v NASA, '24 Houston Center', ADDR='Cape Canaveral', COUNTRY=USA

For more information on the WebFOCUS Developer Studio Command Console, see the Creating Reporting Applications with Developer Studio manual.

**Using the WFMSTART.HTML Launch Page**

The WFMSTART.HTML launch page, found in the /approot/ibisamp directory, can launch any WebFOCUS Maintain application.

You can also use this launch page as a basis for creating your own launch page.

The launch page also includes an entry field for a Google API License. You can embed Google Maps into your Maintain applications, using the Google Maps JavaScript API. To do this, you need to first get an API key. You can refer to the Designing a User Interface for a Web Application With HTML Composer manual for more information on generating a Google Maps API license key. By entering the license key in the sample WFMSTART.HTML launch page, you can pass the API key to your Maintain application, allowing you to incorporate Google Maps onto your forms. You can use an HTML Object to display the map, and JavaScript commands to manipulate it. For more information on attaining the key and manipulating the object, see the documentation found at the Google Maps API website.
Security and Running WebFOCUS Maintain Applications

At run time, your application must collect the appropriate security credentials from end users to access the application and the data.

Suppling DBA Information at Run Time

FOCUS databases enable you to turn on DBA (that is, end users must supply a password to access a data source).

1. Design a web form to collect DBA information when your application starts.
2. Place the DBA password in a special cookie entry in the WF_USER cookie called DBAPASS.
3. If DBAPASS is sent, you can modify your WebFOCUS Servlet scripts to send it to the WebFOCUS server as a SET PASS = &DBAPASS command in the IBISTART.WFS file.

For a view of this technique in action, generate an Update Assist application against a DBA protected Master File description. Make sure you select the Prompt for DBA option at the Navigation step. The resulting navbar HTML file uses this technique.

For more information on Update Assist applications, see the Creating Reporting Applications with Developer Studio manual.

For more information on setting up DBA, see the Describing Data With Graphical Tools or Describing Data With WebFOCUS Language manuals.

Store and Retrieve Credentials With Web Browser Cookies

Information on cookies and storing credentials can be found in the WebFOCUS Security and Administration manual and the Developing Reporting Applications manual.

Customizing Key Functions and JavaScript for WebFOCUS Maintain Applications

You can restrict key functions and include JavaScript functions to be used by all of your WebFOCUS Maintain applications by means of the mntonload.js file. This file is located in \ibi_html\javaassist\ibi\html\maint, and is included in every deployed Maintain application. The functions in mntonload.js are run before the Maintain form is loaded and every time it is refreshed.
Key Restriction

Certain keys and actions, such as the Backspace key, can disconnect a WebFOCUS Maintain application when you use them at certain times within the browser session running the application. You can restrict the Backspace key, function keys, or the right-click context menu when running a WebFOCUS Maintain application by editing the mntonload.js file, located in \ibi_html\javaassist\ibi\html\maint. The default mntonload.js file does not restrict any keys. To customize the behavior of these keyboard actions for your environment, do the following:

- Uncomment the `//BlockKeys();` line near the beginning of the file to restrict the use of the Backspace key and all function keys except F1. For example:

  ```javascript
  function mntOnLoad()
  {
    //To restrict the BackSpace key and all Function keys except F1,
    //uncomment the following line: BlockKeys();
  }
  ....
  ```

- Uncomment the following two lines to restrict the use of the right-click context menu:

  ```javascript
  \document.oncontextmenu=disable_context_menu;
  ....
  \document.onmousedown=click;
  ```

  For example:

  ```javascript
  // To restrict use of Right-Click context menu,
  // uncomment the next line and the last line of this file
  document.oncontextmenu = disable_context_menu;
  function disable_context_menu()
  ....
  // To restrict use of Right-Click context menu, uncomment the
  // following line and the 'document.oncontextmenu=...' line noted above
  document.onmousedown=click;
  ```

  Customized versions of mntonload.js will be saved in a backup directory when you use the Update option when installing a later service pack.

Customized JavaScript Functions

You can include your own JavaScript functions to be used at run time by all of your WebFOCUS Maintain applications. The code can go anywhere inside the body of the mntonload procedure. For it to be performed, you must have the function call in the header.
For example:

```javascript
function mntOnLoad()
{
    //To restrict the BackSpace key and all Function keys except F1,
    //uncomment the following line:
    BlockKeys();
    myExample();
}
...
function myExample()
{
    ...
}
```

**Closing WebFOCUS Maintain Applications**

When an end user closes the browser window containing the WebFOCUS Maintain application, a disconnect is issued.

This way, you do not need to set the server variable idle_session_limit to a relatively low value in order to force a cleanup of agents left in unusable states after exiting a Maintain application.
Designing Applications for Scalability and Performance

This section describes strategies for drafting and implementing an application design that will provide the best flexibility for deployment. The term flexibility includes the ability of the application to perform well, use resources optimally, and provide the business functions desired. Most often, this means spreading applications across different computing resources and tuning different components of the application.

During initial deployment and user population ramp up, certain scalability characteristics will become evident and may require that you move parts of the application to different physical servers.

In this chapter:

- Application Partitioning
- Presentation Logic
- Database Logic
- Miscellaneous Business Logic
- EXEC Command, CALL Command, and System Resources
- Use When Appropriate
- Using the Best Parts
- Persistence Management
- Development Standards
- Tips for Designing Applications

Application Partitioning

In WebFOCUS Maintain applications, application modules can be partitioned (logically or physically separated from each other). The following types of application modules can be characterized and uniquely identified:

- Presentation logic
- Database logic
Often, what logic actually ends up implemented in any of these areas can defy or contradict the original definition of the actual type of module in which it resides. It is important to adhere to partitioning guidelines. The objective of partitioning is to insert separators between different functional parts of the application, so that each of these functional parts can be run-time optimized by a server or servers and shared between functions, where possible. Application partitioning prevents a monolithic platform, which can never be best optimized for presentation logic or database logic.

Presentation logic and database logic pertain to the two main sections of code that make up the Maintain portion of any given application. The assumption is that at run time, these two sections of code will reside and execute on two different physical Maintain servers. The server running the presentation logic will have no access to the DBMS that the application is using. In addition, the server running the presentation logic will have a binding relationship with the web server (or HTTP server) and be optimized in this direction.

Conversely, the server running the database logic will have no access to the web server, but will have a binding relationship with the DBMS. In this way, the two servers can be tuned and appropriately supported with hardware as required.

The EXECed procedures can also run on the database logic platform server. This is often sufficient in terms of performance and resource utilization. However, it is sometimes warranted to replicate the database logic platform server to create a new server dedicated to EXEC/TABLE-based reports. The server used for EXECs can also be used for the real reporting portion of the application, which is more tuned for longer running and large retrieval procedures.

**Presentation Logic**

Presentation logic refers to the Maintain syntax and browser script (usually JavaScript) code that describes and implements the user interface for the application. When you are authoring forms in WebFOCUS Developer Studio, you are:

- Building the actual screen.
- Binding data controls to stacks and other variables.
- Adding user navigation controls.
- Authoring the browser-based validations and animations that occur.
Most of this logic is stored in a resource file (.wfm file), and often all you will see in the code procedure (the .MNT or MAINTAIN file) related to your form is Winform Show. At compile-time, both the .MNT code and the .wfm resource code are built into a run-time procedure (the .FCM, or FOCCOMP file). The actual Maintain code that retrieves or updates data from a database and manipulates that data (or any other non-form logic) should not be implemented in this procedure. These Maintain database commands should be implemented in different procedures that are called or execed from this presentation logic procedure.

The presentation logic procedure must follow the rules of the Maintain compiler and run-time environment. It must have a beginning and end, and any data structures referenced in the form logic or in the CALL or EXEC parameter lists must be described and initialized. In the following presentation logic example, the Getdata and Putdata database logic procedures have the same MAINTAIN FILE MOVIES starting syntax and the same Infer command:

```maintain
MAINTAIN FILE MOVIES
Infer moviestk from movies
Case Top
    Winform_Show
EndCase
Case GetData
    CALL getdata from moviestk into moviestk
EndCase
Case PutData
    CALL putdata from moviestk into moviestk
EndCase
END
```

Presentation logic procedures are always invoked by the submit request of a browser, through WebFOCUS Maintain. All other procedure types are invoked by CALL or EXEC commands from a running Maintain procedure.

**Database Logic**

Database logic, strictly defined, is the code that implements the following Maintain database commands:

- NEXT
- INCLUDE
- UPDATE
- DELETE
Applications typically apply a stack to these commands to load or put back data to and from a database. The stack or stacks are typically visible in the presentation logic because they are bound to visual controls. This enables the user to work with the data in the stacks on a form.

In a partitioned application, the stacks are initialized in both the presentation logic procedures and the database logic procedures, either through INFER or COMPUTE statements. The stack data is then passed back and forth between the two sets of Maintain procedures using the CALL parameter syntax, FROM and INTO.

In normal practice, however, the stack retrieved from the database and the stack presented to the user have usually gone through some amount of transformation, so little similarity remains between them. For example, new columns may be added, the stack may be sorted differently, or it may be copied to a secondary stack for purposes important to the logic of the application. Therefore, there will always be more logic required than simple database commands creating stacks that are then presented to the user. Additional logic must be included between these two actions.

The best place for these non-database commands and logic (which are also not presentation logic) is in the database logic component of the application. This way, the presentation logic procedures must only show forms and send and receive data stacks from the database logic procedures. The database logic procedures take on the role of interfacing with the DBMS, as well as preparing the stacks for presentation and/or re-preparing them for DBMS interaction.

One way to characterize this processing is pre-form stack processing and pre-database stack processing. The stacks must be fully transformed to match the data control being used on the form before being returned to the presentation logic procedure. The stacks may also need to be re-transformed before being addressed back to the DBMS.

**Miscellaneous Business Logic**

This area of an application is what really differentiates one application from another. It also makes generalizations difficult. Normally, the previous notes about pseudo-database logic apply here also. In almost all cases, it is better to call back to the database logic platform from the presentation logic platform for anything other than trivial form navigation. In almost any deployment, the back-end server running the database logic is less busy, and the hardware is often more powerful than the front-end server running the presentation logic. However, if communication performance is not satisfactory, you must account for this in your partitioning and logic placement.
EXEC Command, CALL Command, and System Resources

Unlike the CALL command, which enables you to call WebFOCUS Maintain functions as RPCs, the EXEC command enables you to call FOCEXECs from Maintain. A Maintain procedure can execute any FOCEXEC that returns either a structured answer set or HTML. The data the FOCEXEC returns can be received back into a stack in the calling Maintain procedure.

Since an EXECed procedure is a FOCEXEC and not a Maintain procedure, please note these differences between the CALL and EXEC commands:

- You cannot send an EXEC procedure a Maintain stack as a parameter. Only scalar parameters are permitted. The FOCEXEC treats them like amper variables.

  **Note:** To pass stack members to a FOCEXEC, copy the stack members to scalar variables and pass them as scalars.

- When on the same server, CALL runs on the same server agent, while EXEC starts a new server agent. When run remotely, CALL and EXEC both start new agents.

- The answer set returned by EXEC does not have to include the full logical database path to the column or columns you requested, as does CALL. This enables you to get only the column or columns you want without having a wider stack than is necessary, as is often the case with Maintain calls.

Use When Appropriate

Although EXEC is used for almost any possible reason, its most common use is for populating stacks for combo and list boxes in the presentation logic. EXEC is used with a TABLE request coded in the FOCEXEC called by a remote procedure. This is advantageous because in many cases, TABLE can retrieve a single column of multiple rows of data faster and more efficiently than Maintain. However, the cost of connecting to another agent can sometimes negate the advantages of EXEC/TABLE.

Using EXEC can also make sense in the early stages of development. For example, it may be very easy to create or utilize an existing piece of code with an EXEC to get the application working, or a concept or prototype realized. As you near completion of your application and begin tuning it for ultimate performance, you may find that the cost or attributes of using EXEC (for example, an additional agent, additional connections, and so on) warrants the re-implementation of this logic in Maintain.
Using the Best Parts

WebFOCUS reporting, and Business Intelligence in general, typically involves large record sets and many levels of aggregation, sorting, and joins. Subsequently, these areas of WebFOCUS have been optimized to use the minimum amount of CPU per retrieved record so they can scale up to millions of records. In addition, on MVS, the following have all been implemented in IBM mainframe Assembler to insure the shortest code path: database I/O and navigation, screening, expression evaluation, sorting and summing, and conversion to printable formats.

WebFOCUS Maintain, and transaction systems in general, typically involve small record sets, data validation, navigation logic, and transaction control. Maintain has been designed for applications composed of short transactions using small sets of data. The optimizations developed have been directed to cause many small transactions to run quickly, not to cause a single transaction that uses thousands of records to run quickly. In addition, Maintain has not been optimized to any particular platform. It is written entirely in portable C using an object oriented/layered approach.

When the use of WebFOCUS Maintain (or any transaction tool kit) involves transactions and database activity beyond small records sets, particular care must be made in selecting the best approach to solving the presented business problem. For example, retrieving large record sets can (and should) be implemented using EXEC/TABLE from within WebFOCUS Maintain. In this way, the best software component is used for the right purpose. When updating a database with large record sets, deferred or batch updates should be given preference over purely online systems.

Persistence Management

By default, presentation logic is persistent with WebFOCUS Maintain. This means that when the application processes a Winform_Show command, a formatted HTML screen is returned to the user, and the procedure and processing agent wait, or persist, until the user invokes another submit through the Servlet to the running procedure. The Attach Manager in the server, using the cookie, reattaches the user to the waiting procedure and processing agent, and processing continues. In this way, the waiting procedure and processing agent retain the user context between the different forms or screens of the application user experience. User context means the value of variables and stacks of an application, position in a database, and state of the transaction at any given time.
When a Maintain procedure CALLs or EXECs another procedure on a different server, by default both the connection and remote procedure remain persistent. The calling Maintain waits for a response from the called procedure and continues processing after a completed response. The called procedure is complete when it reaches the EXIT command and returns control to the calling procedure. No communications disconnect is issued. The called procedure is instantly recallable using the persistent connection and the memory resident status of the procedure.

By using the keyword DROP in the CALL or EXEC statement, the connection can be dropped between the calling and the remote called or execed procedure after control has been returned to the calling procedure. In this way, a large number of presentation logic procedures can share a smaller number of database logic procedures on a DBMS platform server.

In certain environments, the cost of new connects or disconnects between servers can be expensive or prohibitive. In these environments and server configurations, the strategy may be to keep the sub-server connected, since the calling server is set to pool its connections to the called server.

Actual usage must be evaluated. For example, do users spend longer periods on certain screens, or do they leave certain screens running when they go to lunch? The possibility or impact of these items must be evaluated before determining the persistence characteristics of an application.

Although a called procedure that has not been dropped maintains connection persistence, by default, the called procedure does not maintain the users context between calls. When the called procedure reaches the EXIT command, it terminates and returns control to the calling procedure. All application variables and stacks, database positions, or transactions in the called procedure are lost. The next time the remote procedure is called, all variables, stacks, and database positions must be reestablished.

To retain the user context in a called procedure, you must add the syntax GOTO END KEEP in a logical place in the called procedure to insure its execution before exiting. When this step is implemented, the called procedure will retain the values in variables and stacks, and retain database positions until it is called again. This approach is often used when the application requires the scrolling of many database records. The user can request more records, and the application retrieves more records, by using the next database position retained by the called procedures context (rather than by retrieving a large answer set prior to a user request for more records). Maintaining user context in the called procedure is also the only way to provide DBMS record locking for those infrequent applications that require this level of transaction control.

In summary, by using the keyword DROP, the application developer is able to control the connection persistence of the CALL or EXEC logic section of the application to best optimize the nature of the application, the available resources, and the actual usage of the application. By using GOTO END KEEP, the developer can control the context persistence of the called procedure.
Note: When evaluating the DROP strategy with any application, the key determining factors are often the details of the actual user behavior, not the code performance or machine or network resource restraints.

Development Standards

Development standards define characteristics that all components of an application will exhibit. Establishing standards helps to ensure that the various components have a common look and feel. Additionally, it assists in continuity where work carried out by one person is taken up and continued by another. Deviations from these standards must be well documented and handled on a case-by-case basis.

Tips for Designing Applications

Ensure that the logic of the application conforms to the following guidelines:

- **Stacks.** It is recommended that you read only as many rows into a stack as are needed for the current transaction, and then clear the stack when its data is no longer needed. This ensures maximum memory availability. You can specify which rows to read into a stack by using the FOR and WHERE phrases of the NEXT command, and you can clear a stack using the STACK CLEAR command.

- **Transactions.** Design your logical transactions to be as small as possible. Also, to prevent bottlenecks, Maintain closes current user transactions before displaying a form to an end user. You must ensure the integrity of the transaction.
Ensuring Transaction Integrity

You are familiar with individual data source operations that insert, update, or delete data source segment instances. However, most applications are concerned with real-world transactions, like transferring funds or fulfilling a sales order, that each require several data source operations. These data source operations may access several data sources, and may be issued from several procedures. We call such a collection of data source operations a logical transaction (it is also known as a logical unit of work).

In this chapter:
- Transaction Integrity Overview
- Why Is Transaction Integrity Important?
- Defining a Transaction
- Evaluating Whether a Transaction Was Successful
- Concurrent Transaction Processing
- Ensuring Transaction Integrity for FOCUS Data Sources
- Ensuring Transaction Integrity for DB2 Data Sources

Transaction Integrity Overview

This topic describes how WebFOCUS Maintain ensures transaction integrity at the application level. At the data source level, each database management system (DBMS) implements transaction integrity in its own way. For more information, see your DBMS vendor documentation for DBMS-specific information. For FOCUS data sources, this DBMS-specific information is presented in Ensuring Transaction Integrity for FOCUS Data Sources on page 368. For DB2, you can find some suggested strategies for writing Maintain transactions to DB2 data sources in Ensuring Transaction Integrity for DB2 Data Sources on page 376. For many other types of data sources, you can also apply the strategies described in Ensuring Transaction Integrity for DB2 Data Sources on page 376, changing DBMS-specific details when necessary.

Example: Describing a Transfer of Funds as a Logical Transaction

A banking application would define a transfer of funds from one account to another as one logical transaction comprising two update operations:
Why Is Transaction Integrity Important?

- Subtracting the funds from the source account (UPDATE Savings FROM SourceAccts).
- Adding the funds to the target account (UPDATE Checking FROM TargetAccts).

**Procedure: How to Process a Logical Transaction**

To process a logical transaction, follow these steps:

1. **DBMS requirements.** The database management system of your data sources (DBMS) may require that you perform some tasks to enable transaction integrity. For more information, see your DBMS vendor documentation for information.

   You can set some native DBMS parameters using the SYS_MGR.FOCSET command. For more information, see the *Maintain Language Reference* manual. You can also set some native DBMS parameters through FOCUS. See your server documentation.

   For FOCUS data sources, you must set the COMMIT server parameter to ON, and issue a USE command to specify which FOCUS Database Server will manage concurrent access to the data source. For more information, see *Ensuring Transaction Integrity for FOCUS Data Sources* on page 368.

2. **Develop the transaction logic.** Code the data source commands and related logic that read from the data sources, write to the data sources, and evaluate the success of each data source command.

3. **Define the transaction boundary.** Code a COMMIT command, and any other supporting commands, to define the transaction boundary. For more information, see *Defining a Transaction* on page 361.

4. **Evaluate the success of the transaction.** Test the FocCurrent transaction variable to determine if the transaction was successfully written to the data source, and then branch accordingly. For more information, see *Evaluating Whether a Transaction Was Successful* on page 365.

**Why Is Transaction Integrity Important?**

The advantage of describing a group of related data source commands as one logical transaction is that the transaction is valid and written to the data source only if all of its component commands are successful. When you attempt to commit a transaction, you are ensured that if part of the transaction fails, none of the transaction will be written to the data source. This is called transaction integrity.

When is transaction integrity important? Whenever a group of commands are related and are only meaningful within the context of the group. In other words, whenever the failure of any one command in the transaction at commit-time would invalidate the entire transaction.
Transaction integrity is an uncompromising proposition: either all of the transaction is written to the data source when you commit it, or all of it is rolled back.

**Example:** Why Transaction Integrity Is Essential to a Bank

Consider a banking application that transfers funds from a savings account to a checking account. If the application successfully subtracts the funds from the savings account, but is interrupted by a system problem before it can add the funds to the checking account, the money would disappear, creating unbalanced bank accounts.

The two update commands (subtracting and adding funds) must be described as parts of a single logical transaction, so that the subtraction and addition updates are not written to the data source independently of each other.

**Defining a Transaction**

You define a logical transaction by issuing a COMMIT or ROLLBACK command following the last data source command of the transaction. For simplicity, the remainder of this topic refers to COMMIT only, but unless stated otherwise, both commands are meant. For example, the beginning of your application is the beginning of its first logical transaction. The data source commands that follow are part of the transaction. When the application issues its first COMMIT command, it marks the end of the first transaction.

The data source commands that follow the first COMMIT become part of the second logical transaction. The next COMMIT to be issued marks the end of the second transaction, and so on.

The COMMIT command defines the boundary of the transaction. All data source commands issued between two COMMIT commands are in the same transaction. This explanation describes the simplest case, in which a transaction exists entirely within a single procedure. When a transaction spans procedures, you have several options for deciding how to define a transaction boundary, as described in *When an Application Ends With an Open Transaction* on page 365.

**Example:** Defining a Simple Transfer of Funds Transaction

For example, transferring money from a savings account to a checking account requires two update commands. If you want to define the transfer, including both updates, as one logical transaction, you could use the following function:

```plaintext
CASE TransferMoney
  UPDATE Savings FROM SourceAccts
  UPDATE Checking FROM TargetAccts
  COMMIT
ENDCASE
```
When Does a Data Source Command Cause a Transaction to Fail?

A data source command can fail for many reasons. For example, an UPDATE command might try to write to a record that never existed because a key was mistyped, or an INCLUDE command might try to add a record that has already been added by another user.

In some cases, when a command fails, you might want to keep the transaction open and simply resolve the problem that caused the command to fail. For example, in the first case of attempting to update a record that does not exist, you might wish to ask the application end user to correctly re-enter the customer code (which is being used as the key of the record). In other cases, you might wish to roll back the entire transaction.

If a data source command fails, it will only cause the logical transaction that contains it to be automatically rolled back in certain circumstances. The deciding factor is when a data source command fails. If a data source command fails when the transaction:

- Is open (that is, when the application issues the data source command), the transaction remains open, and the failed data source command does not become part of the transaction. This means that, if the application later attempts to commit the transaction, because the failed data source command is not part of the transaction, it will not affect the success or failure of the transaction.

  You can evaluate the success of a data source command in an open transaction by testing the value of the FocError system variable immediately after issuing the command. If you wish the failure of the data source command to roll back the transaction, you must issue a ROLLBACK command.

- Is being closed (that is, when the application tries to commit the transaction), the failure of the data source command to be written to the data source causes the transaction to fail, and the entire transaction is automatically rolled back.

Canceling a Transaction

A transaction that is ongoing and has not yet been committed is called an open transaction. To cancel an open transaction, you must issue a ROLLBACK command. ROLLBACK voids any of the data source commands of the transaction that have already been issued so that none of them are written to the data source.

Transactions and Data Source Position

When a logical transaction is committed or rolled back, it resets all position markers in all the data sources that are accessed by the transaction procedures. Resetting a data source position markers points them to the beginning of the data source segment chains.
How Large Should a Transaction Be?

A transaction is at its optimal size when it includes only those data source commands that are mutually dependent upon each other for validity. If you include independent commands in the transaction and one of the independent commands fails when you try to commit the transaction, the dependent group of commands will be needlessly rolled back.

For example, in the following banking transaction that transfers funds from a savings account to a checking account

```plaintext
CASE TransferMoney
  UPDATE Savings FROM SourceAccts
  UPDATE Checking FROM TargetAccts
  COMMIT
ENDCASE
```

you should not add an INCLUDE command to create a new account, since the validity of transferring money from one account to another does not depend upon creating a new account.

Another reason for not extending transactions unnecessarily is that, in a multi-user environment, the longer a transaction takes, the more likely it is to compete for records with transactions submitted by other users. Transaction processing in a multi-user environment is described in Concurrent Transaction Processing on page 366.

Designing Transactions That Span Procedures

Logical transactions can span multiple Maintain procedures. If a Maintain procedure with an open transaction passes control to a WebFOCUS procedure, the open transaction is suspended. When control next passes to a Maintain procedure, the transaction picks up from where it had left off.

When a transaction spans several procedures, you will usually find it easier to define the boundaries of the transaction if you commit it in the highest procedure in the transaction (that is, in the procedure closest to the root procedure). Committing a transaction in a descendant procedure of a complex application, where it is more difficult to track the flow of execution, makes it difficult to determine the transaction's boundaries (that is, to know which data source commands are being included in the transaction).

When a child procedure returns control to its parent procedure, and the child has an open logical transaction, you have two options:

- You can continue the open transaction of a child into the parent procedure when the child returns control to the parent. Simply specify the KEEP option when you return control with the GOTO END command.
You can close the open transaction of the child automatically at the end of the child procedure. By default, Maintain issues an implied COMMIT command to close the open transaction. You can also specify this behavior explicitly by coding the RESET option when you return control with the GOTO END command.

KEEP and RESET are described in the Maintain Getting Started and Maintain Language Reference manuals.

**Example:**  Moving a Transaction Boundary Using GOTO END KEEP

Consider a situation where procedure A calls procedure B, and procedure B then calls procedure C. The entire application contains no COMMIT commands, so the initial logical transaction continues from the root procedure (A) through the descendant procedures (B and C). C and B both return control to their parent procedure using a GOTO END command.

The table below shows how specifying or omitting the KEEP option when procedures B and C return control affects the transaction boundaries of the application (that is, how the choice between KEEP and the implied COMMIT determines where the initial transaction ends, and how many transactions follow).

| C returns to B with... | B returns to A with... | Transaction boundaries (||) |
|-----------------------|-----------------------|-----------------------------|
| KEEP                  | KEEP                  | A-B-C-B-A one transaction   |
| KEEP                  | implied COMMIT        | A-B-C-B || A two transactions|
| implied COMMIT        | KEEP                  | A-B-C || B-A two transactions|
| implied COMMIT        | implied COMMIT        | A-B-C || B || A three transactions|

**Designing Transactions That Span Data Source Types**

If a transaction writes to multiple types of data sources, each database management system (DBMS) evaluates its part of the transaction independently. When a COMMIT command ends the transaction, the success of the COMMIT against each data source type is independent of the success of the COMMIT against the other data source types. This is known as a broadcast commit. If any part of the broadcast commit fails, the value of FocCurrent is not zero.
For example, if you issue a Maintain procedure against the FOCUS data sources Employee and JobFile, and a DB2 data source named Salary, the success or failure of the COMMIT against Salary is independent of its success against Employee and JobFile. It is possible for it to be successful against Salary and write that part of the transaction, while being unsuccessful against Employee and JobFile and roll back that part of the transaction. Because it is unsuccessful against Employee and JobFile, the value of FocCurrent is not zero.

**Designing Transactions in Multi-Server Applications**

In an application that spans multiple WebFOCUS Servers, the server defines the maximum scope of a logical transaction. No transaction boundary can extend beyond a WebFOCUS Server. If one of your applications spans several servers, protect its transaction boundaries by ensuring that:

- All of the procedures of the application that read and write to a given data source reside on the same WebFOCUS Server.

- In each of the transactions of the application that span multiple procedures, all of the transaction’s procedures that read and write to data sources reside on the same WebFOCUS Server.

If a procedure with an open transaction calls another procedure that resides on a different WebFOCUS Server, and the situation violates either of the previous restrictions, the data source commands on the new server will comprise a new transaction. When control returns to the calling procedure on the original server, the original open transaction continues from where it had left off.

**When an Application Ends With an Open Transaction**

If an application terminates while a logical transaction is still open, Maintain issues an implied COMMIT command to close the open transaction, ensuring that any data source commands issued after the last explicit COMMIT are accounted for. The only exception is if your WebFOCUS Maintain session abnormally terminates: Maintain does not issue the implied COMMIT, and any remaining uncommitted data source commands are rolled back.

**Evaluating Whether a Transaction Was Successful**

When you close a transaction by issuing a COMMIT or ROLLBACK command, you must determine whether the command was successful. If a COMMIT command is successful, then the transaction it closes has been successfully written to the data source. If a ROLLBACK command is successful, then the transaction it closes has been successfully rolled back.
The system variable FocCurrent provides the return code of the most recently issued COMMIT or ROLLBACK command. By testing the value of FocCurrent immediately following a COMMIT or ROLLBACK command, you can determine if the transaction was successfully committed or rolled back. If the value of FocCurrent is:

- Zero, the command was successful.
- Not zero, the command was unsuccessful.

FocCurrent is global to a procedure. If you want a given value of FocCurrent to be available in a different procedure, you must explicitly pass it as an argument to that procedure.

**Example: Evaluating the Success of a Transaction**

The following function commits a transaction to a data source. If the transaction is unsuccessful, the application invokes another function that writes to a log and then begins a new transaction. The FocCurrent line evaluates the success of the transaction:

```plaintext
CASE TransferMoney
    UPDATE AcctBalance FROM SourceAccts
    UPDATE AcctBalance FROM TargetAccts
    COMMIT
    IF FocCurrent NE 0 THEN PERFORM BadTransfer
ENDCASE
```

**Concurrent Transaction Processing**

Several applications or users often need to share the same data source. This sharing can lead to problems if they try to access a record concurrently, that is, if they try to process the same data source record at the same time.

To ensure the integrity of a data source, concurrent transactions must execute as if they were isolated from each other. The changes of one transaction to a data source must be concealed from all other transactions until that transaction is committed. To do otherwise, runs the risk of open transactions being exposed to interim inconsistent images of the data source, and consequently, corrupting the data source.

To prevent users from corrupting the data in this way, the database management system (DBMS) must coordinate concurrent access. There are many strategies for doing this. No matter which type of data source you use, WebFOCUS Maintain respects your DBMS concurrency strategy and lets it coordinate access to its own data sources.
For more information about how your DBMS handles concurrent access, see your DBMS documentation. For FOCUS data sources, this information is presented in *Ensuring Transaction Integrity for FOCUS Data Sources* on page 368. For DB2, you can find some suggested strategies for writing Maintain transactions to DB2 data sources in *Ensuring Transaction Integrity for DB2 Data Sources* on page 376. For many other types of data sources, you can also apply the strategies described in *Ensuring Transaction Integrity for DB2 Data Sources* on page 376, changing DBMS-specific details when necessary.

**Example: Why Concurrent Access to a Data Source Must Be Managed Carefully**

Consider the following two applications that access the Employee data source:

- The Promotion application reads a list of employees who have received promotions, and updates their job codes to correspond to their new positions.
- The Salary application, run once at the beginning of each year, checks every employee job code and gives each employee an annual raise based on his or her job title. For example, assistant managers (job code A15) will earn $30,000 in the new year, and managers (A16) will earn $40,000.

Joan Irving is an assistant manager. Consider what happens when these two applications try to access and update the same record at the same time, without any coordination:

1. The Promotion application reads the record of Irving and, based on information in a transaction data source, indicates that she has been promoted to manager, and computes her new job code (A16).
2. The Salary application reads record of Irving and, based on her job code in the data source (A15), computes her new salary ($30,000).
3. The Promotion application writes the new job code (A16) to the data source.
4. The Salary application writes the new salary ($30,000) to the data source.

Remember the earlier business rule (assistant managers earn $30,000, managers earn $40,000). Because two applications accessed the same record at the same time without any coordination, the rule has been broken (Joan Irving has a manager job code but the salary of an assistant manager). The data source has become internally inconsistent.
Ensuring Transaction Integrity for FOCUS Data Sources

Each database management system (DBMS) supports transaction integrity in its own way. The FOCUS DBMS manages concurrent access to FOCUS data sources using the FOCUS Database Server, and uses certain commands to identify transaction integrity attributes. The FOCUS Database Server was formerly known as a sink machine, or Simultaneous Usage (SU) facility on some platforms.

To ensure transaction integrity for FOCUS data sources, perform the following tasks:

- **Install the FOCUS Database Server.** When installing each WebFOCUS Server that will host FOCUS data sources, select the FOCUS Database Server option.

- **Set COMMIT.** Set the COMMIT server parameter to ON. This enables the COMMIT and ROLLBACK commands for FOCUS data sources, and enables the use of the FOCUS Database Server. For more information, see `Setting COMMIT` on page 368.

- **Select which segments will be verified for changes.** Set the PATHCHECK server parameter to specify the type of segments for which the FOCUS Database Server will verify change. This is optional: you can accept the default setting. For more information, see `Selecting Which Segments Will Be Verified for Changes` on page 371.

- **Identify the FOCUS Database Server.** Identify which FOCUS Database Server will manage concurrent access to each FOCUS data source. For more information, see `Identifying the FOCUS Database Server` on page 372.

- **Start the FOCUS Database Server.** Under Windows and UNIX, when you start the WebFOCUS Server, it automatically starts the FOCUS Database Server. When you stop the WebFOCUS Server, it automatically stops the FOCUS Database Server. For information about starting and stopping the FOCUS Database Server under MVS and OS/390, see the `Simultaneous Usage Reference Manual, TSO Version` manual.

### Setting COMMIT

You must set the COMMIT server parameter to ON before using the COMMIT and ROLLBACK commands for FOCUS data sources, and before using the FOCUS Database Server. You must set COMMIT on all WebFOCUS Servers hosting procedures that read or write to FOCUS data sources in a logical transaction. In most applications, this will mean setting COMMIT on all WebFOCUS Servers that host procedures with data source commands.

You can set COMMIT:

- Comprehensively for all users on a WebFOCUS Server. Issue the SET COMMIT command in the server’s global profile (EDASPROF).
Comprehensively for a group of users on a WebFOCUS Server. Issue the SET COMMIT command in one or more of the group profiles of the server. Group profiles are supported under UNIX, OS/390, and MVS.

Individually for each user on a WebFOCUS Server. Issue the SET COMMIT command in one or more of the group profiles of the server (PROFILE). The user in this case is the user account that launches the application.

If you set COMMIT in a user profile or group profile, you must set it in the profile of the user or group that runs the application.

You can also set COMMIT directly from a Maintain procedure.

**Syntax:**

**How to Set COMMIT**

The COMMIT server parameter enables transaction integrity for FOCUS data sources. To set COMMIT, issue the SET COMMIT command in a WebFOCUS Server global profile, or in one or more of its user or group profiles, using the following syntax:

```sql
SET COMMIT={ON|OFF}
```

To set COMMIT in a Maintain procedure, use the following syntax

```sql
SYS_MGR.FOCSET("COMMIT"{"ON|}"
```

where:

- **ON**
  
  Enables the COMMIT and ROLLBACK commands for use with FOCUS data sources, and enables the use of the FOCUS Database Server to ensure transaction integrity.

- **OFF**
  
  Disables the COMMIT and ROLLBACK commands for use with FOCUS data sources, and disables the use of the FOCUS Database Server to ensure transaction integrity. OFF is the default value.

**Sharing Access to FOCUS Data Sources**

The FOCUS DBMS ensures transaction integrity when multiple users are trying to access the same data source concurrently. If you are processing a transaction and, in the interval between beginning your transaction and completing it, the segments updated by your application have been changed and committed to the data source by another user, Maintain will roll back your transaction. This coordination is performed by the FOCUS Database Server. You can test if your transaction was rolled back by checking the value of the FocCurrent transaction variable, and then branch accordingly.
This strategy, where FOCUS verifies that the records to which you wish to write have not been written to by another user in the interim, is called change verification. It enables many users to share write access to a data source, and grants update privileges for a given record to the first user that attempts the update.

Change verification takes advantage of the fact that two users rarely try to update the same record at the same time. Some DBMSs use strategies that lock out all but one user. Others grant update privileges to the first user that retrieves a record, even if he or she is the last one ready to update it, resulting in a performance bottleneck. In contrast, the FOCUS DBMS strategy of change verification enables the maximum number of users to access the same data concurrently, and makes it possible to write the maximum number of transactions in the shortest time. The FOCUS Database Server and change verification strategy are designed for high-performance transaction processing.

How the FOCUS Database Server and Change-verification Work

The change-verification strategy of the FOCUS Database Server is an extension of basic transaction processing. Each application user who accesses the FOCUS Database Server is known as a client. To ensure transaction integrity follow this simple change-verify protocol:

1. As always, use the NEXT or MATCH commands to retrieve the data source records you need for the current transaction. When the application issues these commands, the server sends the application a private client copy of the records.

   **Note:** Do not retrieve data from a data source by running a report procedure. The FOCUS Database Server does not check this data for changes when you attempt to commit a transaction.

2. When the application issues a data source write command (such as INCLUDE, UPDATE, REVISE, or DELETE) against the retrieved records, it updates its private copy of the records.

3. When the application issues a COMMIT command to indicate the end of the transaction, the application session sends a log of the transaction back to the server. The server now checks to see if any of the segments that the transaction changed have, in the interim, been changed and committed to the data source by other clients, and if any segments that the transaction added have, in the interim, been added by other clients. You can customize which segments the FOCUS Database Server checks for changes by setting the PATHCHECK server parameter, as described in *Selecting Which Segments Will Be Verified for Changes* on page 371.

The server takes one of the following actions:
- **No conflict.** If none of the records has been changed or added in the interim, then the transaction is consistent with the current state of the data source. The server writes the transaction to the data source and sets the FocCurrent transaction variable of the application to zero to confirm the update.

- **Conflict.** If any records have been changed in the interim, then the transaction might be inconsistent with the current state of the data source. The server ignores the transaction changes to the data source, rolling back the transaction, and alerts the application by setting FocCurrent to a nonzero number.

4. The application evaluates FocCurrent and branches to the appropriate function.

### Selecting Which Segments Will Be Verified for Changes

When you use a FOCUS Database Server, you can customize the change verification process by defining the segments for which the FOCUS Database Server will verify changes. You define this, using the server parameter described in *How to Set PATHCHECK* on page 371.

You can choose between:

- **All segments in the path.** The FOCUS Database Server verifies that all segments in the path extending from the root segment to the target segment have not been changed and committed in the interim by other users.

- **Modified segments only.** The FOCUS Database Server determines which segments you are updating or deleting, and verifies that those segments have not been changed and committed in the interim by other users.

You can set PATHCHECK for each FOCUS Database Server, which affects all applications that access FOCUS data sources managed by that FOCUS Database Server. To set it under:

- **Windows and UNIX,** issue the SET PATHCHECK command in the batch file (EDASTART.BAT) that starts the FOCUS Database Server.

- **OS/390 and MVS,** issue the SET PATHCHECK command in the FOCUS Database Server profile (HLIPROF).

### Syntax: How to Set PATHCHECK

The PATHCHECK server parameter defines which segments the FOCUS Database Server will check for changes. To set PATHCHECK, issue the SET PATHCHECK command in the batch file that starts the FOCUS Database Server, using the following syntax

```
SET PATHCHECK={ON|OFF}
```
where:

**ON**

Instructs the FOCUS Database Server to verify that all segments in the path extending from the root segment to the target segment have not been changed and committed in the interim by other users. This is the default for OS/390 and MVS.

**OFF**

Instructs the FOCUS Database Server to check only segments that the current transaction has updated or deleted, and verify that those segments have not been changed and committed in the interim by other users. This is the default for Windows and UNIX.

**Identifying the FOCUS Database Server**

To identify which FOCUS Database Server will manage access to a given FOCUS data source, you must issue a command that associates the server with the data source, as described in *How to Identify a FOCUS Database Server With USE* on page 372.

You can issue this command:

- Comprehensively, for all users on a WebFOCUS Server. Issue the USE command in the server’s profile (EDASPROF).
- Individually, for each user on a WebFOCUS Server. Issue the USE command in the user server profile (PROFILE). The user in this case is the user account that launches the application.

**Syntax: How to Identify a FOCUS Database Server With USE**

For each FOCUS data source that will be managed by a FOCUS Database Server, you must associate the data source with the server by issuing a USE command in a WebFOCUS Server profile. The USE command syntax is:

```
USE
  datafile ON server_id
  [datafile ON server_id]
  
  
  
END
```

where:

- **datafile**

  Is the file specification of a data source to be managed by the FOCUS Database Server.
server_id

Under Windows and UNIX is the node name of the FOCUS Database Server, as defined in the FOCUS Database Server node block of the Data Server configuration file.

Under OS/390 and MVS is the ddname of the communication dataset that points to the FOCUS Database Server job.

If you wish, you can identify multiple data source/server pairs in one USE command.

Using Report Procedures and a FOCUS Database Server

When a FOCUS Database Server manages access to a FOCUS data source, each logical transaction that accesses that data source works with its own private copy of the data source’s records. This ensures that the transaction sees a consistent image of the data source that is isolated from changes being attempted by other users.

WebFOCUS procedures, such as report procedures, are not part of a logical transaction; when control passes from a Maintain procedure to a WebFOCUS procedure, the open transaction is suspended for the duration of the WebFOCUS procedure. Therefore, if the WebFOCUS procedure reports against a FOCUS data source, it accesses the live data source. It does not open the private copy of the transaction. Changes made by the open transaction are not seen by the report, and changes committed by other users since the open transaction began are seen by the report, though not necessarily by the open transaction.

For similar reasons, you should not use a report procedure to retrieve data for use in a transaction. The FOCUS Database Server does not check this data for changes when you attempt to commit a transaction. Always use the NEXT or MATCH commands to retrieve transaction data.

If you wish to deploy WebFOCUS procedures containing report requests to a WebFOCUS Server that also hosts Maintain procedures, you must represent the server as two different outbound nodes, and deploy WebFOCUS reporting procedures to one node and Maintain procedures to the other node, as described in Accessing Report Procedures When Using a FOCUS Database Server on page 373. Otherwise, the WebFOCUS procedures may interfere with your transaction logic.

Accessing Report Procedures When Using a FOCUS Database Server

If you are using a FOCUS Database Server and you wish to access Maintain procedures and WebFOCUS report procedures that are located on the same WebFOCUS Server (referred to here as the target server), you must:

1. Represent the target server as two different outbound nodes. Represent it this way to:

   - WebFOCUS Developer Studio.
Each WebFOCUS Server that executes Maintain procedures or WebFOCUS procedures that are located on the target server.

This requirement also applies to the target server itself: if it executes Maintain procedures or WebFOCUS procedures that are deployed on itself, it must be represented to itself as two outbound nodes.

2. Of the procedures that you will be deploying to the target server, deploy the WebFOCUS report procedures to one of these outbound nodes, and deploy the Maintain procedures to the other outbound node.

This is necessary because deploying both types of procedures to the same outbound node can cause report logic to corrupt transaction integrity.

**Procedure: How to Access Maintain and WebFOCUS Report Procedures on the Same Server**

In an application that uses a FOCUS Database Server, if you want to deploy to the Maintain procedures of the application and WebFOCUS report procedures on the same WebFOCUS Server (referred to here as the target server):

1. Represent the target server to WebFOCUS Developer Studio as two outbound nodes that have different remote server names but the same protocol options (for the TCP/IP protocol, this means specifying the two nodes with the same IP address, port number, and compression setting).

   For more information, see the Creating Reporting Applications With Developer Studio manual.

2. Represent the target server to WebFOCUS Servers as two outbound nodes. Perform this step for each WebFOCUS Server that executes Maintain procedures or WebFOCUS procedures that are deployed on the target server. This requirement also applies to the target server itself: if it executes Maintain procedures or WebFOCUS procedures that are deployed on itself, it must be represented to itself as two outbound nodes.

   As in step 1, define the two outbound nodes as having different remote server names but the same protocol options (for the TCP/IP protocol, this means specifying the same IP address, port number, and compression setting). You can do this by copying the target server node block, as described in the WebFOCUS Server Communications Configuration File Location and Name on page 375, and paste it just below the end of the original block. The node block begins with the NODE keyword and continues through the END keyword. Edit the pasted block to provide a new eight-character node name, but leave the block’s other values unchanged.
If the WebFOCUS Server is the target server, then copy the target server node block from WebFOCUS Developer Studio odin.cfg file (located in the webfocusdesktop_root_dir\srv80\wfs\etc directory), and paste it twice, into the communications configuration file of the target server. Edit the second pasted block to provide a new eight-character node name, but leave the other values of the block unchanged.

For information about the name and location of the communications configuration server file, see Communications Configuration File Location and Name on page 375.

3. Deploy the application.

For more information about deploying applications, see Partitioning and Deploying Project Files in the Creating Reporting Applications With Developer Studio manual.

Reference: Communications Configuration File Location and Name

The WebFOCUS Server communications configuration file under:

- **Windows and UNIX** is odin.cfg, and resides in the etc subdirectory of the Data Server configuration directory. Under UNIX, odin.cfg is in the $EDACONF/etc directory. Under Windows, you can find the name of the configuration directory in the environment variable EDACONF.

When you install and configure a WebFOCUS Server, the configuration directory defaults to $HOME/ibi/srv80/server_instance under UNIX, and home\ibi\srv80\server_instance under Windows. HOME is an environment variable whose value in this context is the name of the home directory of the user account that installed the WebFOCUS Server. server_instance is the name of a server configuration directory (there is one directory per server instance. The convention is to name this directory wfs for a WebFOCUS Server, and wfm for a WebFOCUS Maintain Application Server).

- **OS/390 and MVS** is allocated to ddname EDACSG in the server startup JCL.

Sharing Data Sources With Legacy MODIFY Applications

A FOCUS data source being managed by a FOCUS Database Server can be accessed by both WebFOCUS Maintain applications and legacy MODIFY applications. Note that while MODIFY allows creating records with duplicate keys, WebFOCUS Maintain does not support FOCUS data sources that have duplicate keys.
Ensuring Transaction Integrity for DB2 Data Sources

DB2 ensures transaction integrity by locking data source rows when they are read. The behavior of a lock depends on the isolation level of a transaction. The techniques suggested here for WebFOCUS Maintain applications all use an isolation level of repeatable read. Repeatable read involves a trade-off: it ensures absolute transaction integrity, but it can prevent other users from accessing a row for long periods of time, creating performance bottlenecks.

Under repeatable read, a row is locked when it is retrieved from the data source, and is released when the transaction that retrieved the row is either committed to the data source or rolled back. A Maintain DB2 transaction is committed or rolled back each time a WebFOCUS Maintain application issues a COMMIT or ROLLBACK command. You explicitly code COMMIT and ROLLBACK commands in your WebFOCUS Maintain application. In some circumstances the application may also issue these commands implicitly, as described in Designing Transactions That Span Procedures on page 363, and in When an Application Ends With an Open Transaction on page 365.

We recommend two strategies for writing transactions to DB2 data sources:

- **Using transaction locking to manage DB2 row locks.** This locks each row for the duration of the transaction, from the time a row is retrieved, until the transaction is committed. In effect, it relies on DB2 to ensure transaction integrity. This is simpler to code, but keeps rows locked for a longer period of time. This is the preferred strategy, unless the duration of its locks interferes excessively with your data source concurrency requirements.

- **Using change verification to manage DB2 row locks.** This locks each row while it is being retrieved, releases the lock, and then relocks the row shortly before writing it to the data source. This technique ensures transaction integrity by verifying, before writing each row, that the row has not been changed by other users in the interim. This is more complex to code, but locks rows for a shorter period of time, increasing data availability.

While these strategies are described for use with DB2 data sources, you can also apply them to transactions against other kinds of data sources, changing DBMS-specific details when necessary.

**Reference:** How WebFOCUS Maintain DB2 Logic Differs From Other IBI Products

If you are familiar with using the DB2 Data Adapter with Information Builders products other than WebFOCUS Maintain, note that WebFOCUS Maintain works with DB2 a bit differently:

- Maintain enables you to issue COMMIT and ROLLBACK commands explicitly. It also issues them implicitly in certain situations, as described in Designing Transactions That Span Procedures on page 363, and in When an Application Ends With an Open Transaction on page 365.
Maintain does not support the SQL DB2 SET AUTOCOMMIT command to control automatic commits.

Because Maintain works on sets of rows, the DB2 Data Adapter does not automatically generate change verification logic.

Using Transaction Locking to Manage DB2 Row Locks

You can use the transaction locking strategy to manage DB2 row locks in WebFOCUS Maintain applications. While this strategy is described for use with DB2 data sources, you can also apply it to transactions against other kinds of data sources, changing DBMS-specific details when necessary. When using transaction locking, your application locks each row with an isolation level of repeatable read for the duration of the transaction, from the time it retrieves the row, until the time it commits or rolls back the transaction.

The following illustration shows the duration of connections, threads, and logical transactions (also known as logical units of work) when you use this strategy:
If your applications are small in scope, comprising only a single procedure, the duration of connections, threads, and logical transactions would look like this:

Compared to change verification, transaction locking is simpler to code, but keeps rows locked for a longer period of time. This may cause other users to experience time outs, in which case DB2 will return a -911 or -904 SQL code. You can mitigate the effect of row locking by:

- Keeping the size of the transaction small, making it less likely that another user will encounter a row locked by your transaction.

- Implementing the change verification strategy described in *Using Change Verification to Manage DB2 Row Locks* on page 379.

- Having user applications check for a locked condition when retrieving rows, and upon encountering a lock, re-issuing the retrieval request a specified number of times in a loop. If the user application exceeds the specified number of attempts, have it display a message to the user indicating that the row is in use, and suggesting that the user try again later.

- Using standard database administration techniques, such as report scheduling, tablespace management, and data warehousing.
Procedure: How to Implement Transaction Locking for DB2

To implement the transaction locking strategy for managing DB2 row locks in WebFOCUS Maintain applications, bind the DB2 Data Adapter plan with an isolation level of repeatable read. The isolation level is a DB2 Data Adapter installation BIND PLAN parameter. In your WebFOCUS Maintain application:

1. **Read the rows.** Retrieve all required rows. Retrieval locks the rows with an isolation level of repeatable read.

2. **Write the transaction to the data source.** Apply the updates of the transaction to the data source.

3. **Be sure to terminate called procedures correctly.** If a Maintain procedure calls another Maintain procedure within the scope of a transaction, the called procedure must return control using the GOTO END KEEP command. For more information about GOTO END KEEP, see Designing Transactions That Span Procedures on page 363.

   **Caution:** If any called procedure within the scope of a transaction returns control without GOTO END KEEP, Maintain issues an implied COMMIT command, releasing all row locks and making the application vulnerable to updates by other users. Be sure to return control using GOTO END KEEP; otherwise, code each transaction within a single procedure, so that the scope of each transaction does not extend beyond one procedure, or use the change verification strategy described in Using Change Verification to Manage DB2 Row Locks on page 379.

4. **Close the transaction.** When the transaction is complete, close it by issuing a COMMIT or ROLLBACK command. The COMMIT or ROLLBACK command releases all row locks.

Using Change Verification to Manage DB2 Row Locks

You can use the change verification strategy to manage DB2 row locks in WebFOCUS Maintain applications. While this strategy is described for use with DB2 data sources, you can also apply it to transactions against other kinds of data sources by changing DBMS-specific details when necessary.

When using change verification, your application retrieves all needed rows into a stack, locking them in the process; releases the locks after retrieval; and then performs all updates against the stack (not against the data source). This enables you to work with the data in the stack as long as necessary without preventing other users from accessing the data source. When you are ready to close the transaction, retrieve the original rows from the data source again, relocking them in the process. Then, compare their current values in the data source to their original values when you first retrieved them, and write the transaction to the data source if the values are the same, that is, if the rows have not been changed by other users in the interim.
Change verification enables the maximum number of users to access the same data concurrently, and makes it possible to write the maximum number of transactions in the shortest time. It is able to do this because it is an optimistic locking protocol, that is, it is optimized for the most common situation, in which at any moment, at most one user will attempt to update a given row. Compared to transaction locking, this is more complex to code, but locks rows for less time, increasing data availability.

The following illustration shows the duration of connections, threads, and logical transactions when you use this strategy for DB2 data sources:

**Procedure**: How to Implement Change Verification for DB2

To implement the change verification strategy for managing DB2 row locks in WebFOCUS Maintain applications, bind the DB2 Data Adapter plan with an isolation level of repeatable read. The isolation level is a DB2 Data Adapter installation BIND PLAN parameter. In your WebFOCUS Maintain application:

1. **Read the rows**. Retrieve all required rows into a stack (for example, Stack1). Retrieval locks the rows with an isolation level of repeatable read.
2. **Free the row locks.** Issue a ROLLBACK command immediately following retrieval in order to release all row locks.

3. **Copy the stack.** Make a copy of the stack (for example, Stack2). You will use this copy later when checking for changes.

4. **Write the transaction to the stack.** Apply the updates of the transaction to the rows in the original stack (Stack1).

5. **Read the rows again.** Retrieve the rows of the transaction from the data source into a new stack (for example, Stack3). Retrieval relocks the rows with an isolation level of repeatable read.

6. **Verify changes.** Compare the original data source values in the copy of the original stack (Stack2) to the current data source values (Stack3) to verify that other users have not changed these rows in the interim.

7. **Write the transaction to the data source.** If any of these rows have been changed in the data source by another user, you can roll back the transaction or take some other action, as your application logic requires. If none of the rows in the transaction have been changed by other users in the interim, your application can apply the transaction updates to the data source, and issue a COMMIT command to commit the transaction.

   The COMMIT or ROLLBACK command releases all row locks.
WebFOCUS Maintain contains tools that help you debug your applications: WebFOCUS Maintain Application Debugger, Type on EDAPRINT command, and Maintain Statement trace.

**In this chapter:**
- WebFOCUS Maintain Application Debugger
- Type on EDAPRINT
- Maintain Statement Trace
- MNTCON PERFORMANCE_ANALYSIS
- Additional Trace Settings

**WebFOCUS Maintain Application Debugger**

The WebFOCUS Maintain Application Debugger allows you to step through your Maintain application while running on the web to assist in identifying and correcting problems encountered at run time. It is a useful debugging tool that can be enabled during the development cycle or when troubleshooting a problem in a production environment.

A Help feature is available through the debugging tool to provide specific instruction on its use.

**Syntax:** **How to Invoke the WebFOCUS Maintain Application Debugger**

The following command can be used to run your WebFOCUS Maintain application using the Maintain Application Debugger:

```
MNTCON APPDEBUG EX procedure
```

where:

- **procedure**
  Is a Maintain procedure with an .mnt extension.
This command is available from the default WebFOCUS Maintain launch form, WFMSTART.HTML by choosing the option, *Execute a procedure with Maintain Application Debugger* from the drop-down list for Invocation Type.

You can also add this command to your own launch form or embed it in a URL call.

This is the most flexible way to launch the debugger, as it will immediately launch the application using the debugger without requiring a separate compile step, and without leaving a compiled module (FCM or FOCCOMP) behind.

**Reference:** Alternate Methods of Invoking the Maintain Application Debugger

Other methods used to invoke the Maintain Application Debugger:

- Add the following line to the server edasprof.prf profile file
  
  ```
  MNTCON APPDEBUG
  ```
  
  and run the application with the syntax:
  
  ```
  MNTCON EX procedure
  ```

- Compile your procedure(s) using the command
  
  ```
  MNTCON APPDEBUG COMPILE procedure
  ```
  
  and run the compiled object with the command:
  
  ```
  MNTCON APPDEBUG RUN procedurename
  ```

**Debugger Screen**

The Maintain Application Debugger screen consists of the following components:
Debugging Your Application

The following example shows how to use the debugger.

**Example:** Loading the Application Into the Debugger

This example describes how to load the application into the debugger.

1. Start the debugger, using one of the methods described in *How to Invoke the WebFOCUS Maintain Application Debugger* on page 383.

   The main file appears in the File Frame. The color green indicates that a line is commented. A blue line is the current line where the debugger is paused. This line is not yet executed.

2. To run the application up to the end or to the nearest break point, click *Run*.

3. You can run your application either line by line or by setting and using break points.
   a. To execute a line:

      - Execute the current line and click *Step Over* to advance to the next line.
      - Execute the current line and stop inside another case (if the current line contains a call to another case). Next, click *Step Over*. 
b. To set break points:

- Click the line where you want to set the break point. When first clicked, the line is highlighted in a gray frame.
- Click Set Break Point or press F9. The line changes to red.
- To deselect a break point, select the line again and, once more, click Set Break Point or press F9.

**Note:** If you try to set a break point to a commented line or to an empty line, you will see the message *Cannot set break point.*

4. To terminate the debugger, click Exit, which accesses the launch form.

**Tracking Variables**

**Global variables and Stacks.** When you change the Category in the View Data Combo Box, the corresponding variable group appears in the Variables Frame. If the variable is not set, its value displays as #empty. When viewing a stack, you can see FocCount and FocIndex and the current stack position. To change the stack position use the first, next, prev and last navigation links.

**Changing variable values.** Currently it is possible to change values only for Global Variables. To change the value, click the variable value in the Variable Frame, enter a new value, and click Enter or just remove focus from the current control. The new value turns red.

**Note:** First the variable is set, then the current line is executed. At this point, the variable is changed on the screen only. It is not yet set inside the application (or Maintain server). It is set only when the debugger connects to the server, that is, when you click Step Over or when you change the current file by using the Select File: Combo Box.

If the value entered cannot be set (is not valid), the alert message, Cannot set value displays and the variable value is shown as 0.

**Navigation Control**

The Select File ComboBox lists the name of the Maintain procedure that is currently running, and the associated Winform and import modules, if any. The Select Case: Combo Box contains a list of all cases of the current file.
Procedure: How to Select a File and Find a Case

1. To change the current file, select another file from this Combo Box. The debugger will go to the Maintain server and load the selected file into the File Frame.

   At this time, running the Winform file itself with the debugging functions (setting break points, and so on) is not supported.

   External procedures and remote Call procedures are not listed here. They run as part of the main application and data is returned to the calling procedure and available to the debugger. However, the debugging functions are not available for the remote procedures themselves.

2. To go to a Case, select a name from Select Case: Combo Box. The File Frame scrolls to the selected case and the beginning of the case is highlighted in blue-green.

Reference: Error Messages and Type Statements

If a non-critical run time error occurs, an alert message pops up when the next debugger screen displays.

Type statements are seen only at the end of the application execution in the case of non-Winform applications, or after the Winform Show command in the case of applications with Winforms.

Reference: Usage Notes: Debugger Limitations

- Procedure lines greater than 70 characters are truncated in the File Frame view.
- Winform files cannot be run with debugging options. The code can be viewed in the File Frame.
- Remote Calls and external procedures are not listed in the Select Files Combo Box and cannot be run with debugging functions. They run as usual when encountered in the application code, and the variables and errors returned are available to the main calling procedure in the debugger.

Type on EDAPRINT

The TYPE ON EDAPRINT command will allow a Type statement within a Maintain procedure to be sent to the EDAPRINT.LOG file instead of to the browser. It will append each TYPE ON EDAPRINT message to the log so that previous statements are not lost. These statements can be used as a debugging tool to verify whether or not certain parts of an application are being executed, to track variable values, and to monitor user activities without stopping the server or generating a large trace file.
Example: Using Type on EDAPRINT

The following code

```
MAINTAIN
  type on edaprint "first to edaprint";
  type "hello world! from user1";
  type on EDAPRINT "hello - send to edaprint";
END
```

generates the following in the EDAPRINT.LOG file:

```
11/20/2002 13:53:28 Pid=00001528 MntType: first to edaprint
11/20/2002 13:53:28 Pid=00001528 MntType: hello- send to edaprint
```

Maintain Statement Trace

Maintain application developers can trace the logic flow of Maintain procedures and measure CPU and memory usage at the procedure, case, and statement level with the following server trace options:

- SET TRACEON=MNTSTMT produces basic Maintain trace information such as time of day, file name, procedure name, file number, line number, statement, milliseconds and Maintain memory.

- SET TRACEON=MNTPERF (used in addition to SET TRACEON=MNTSTMT) causes tracing to include agent memory and CPU performance statistics.

Note: There will be a significant increase in CPU overhead when MNTPERF is in effect.

Trace operation codes generated at the compile stage determine the Maintain statements for which the above information is provided.

Trace records will be produced when a statement completes. If a Maintain procedure fails, an error message (3850) will print the last traced line.

There are two basic phases in the process of generating trace output for Maintain procedure at run time:

1. Compiling Maintain procedures.
2. Enabling server tracing and setting Maintain trace levels.
Compiling Maintain Procedures

Trace operation codes are generated for the procedure at the time of the compile. There are two levels of statement trace detail available: basic and verbose. The level of detail in the trace output at run time depends on the type of compile statement used.

- A basic compile command, such as a MNTCON COMPILE proname statement, will cause trace lines to be issued for each Case, Endcase and Call and Exec statement.
- A DEBUG option for verbose trace statement information, available at either compile time or as a server setting or execution command, will generate trace output for every Maintain statement.

You can set Compiling and Compiling with Debug (verbose statement trace) at deployment time from within Developer Studio with settings in the Property option of a Deploy Scenario folder. For more information, see the Creating Reporting Applications With Developer Studio manual.

You can also use the following Maintain language commands:

- MNTCON COMPILE procname will produce trace op-codes for each CASE entry, ENDCASE, CALL and EXEC statement.
- MNTCON DEBUG COMPILE procname will produce trace op-codes for every Maintain statement, such as COMPUTE, IF, FOR ALL NEXT.
- MNTCON EX procname or MNTCON DEBUG EX procname operate like their COMPILE counterparts above, but will generate trace output without requiring an explicit compile step, if the appropriate trace settings are enabled on the server. (A MNTCON EX procname does a compile behind the scenes and runs the procedure without writing the compiled version to disk).

**Example:** MNTCON COMPILE MNTCAR1 Using MNTSTMT

Note lines for Maintain statements Case and Endcase, and Call:

12.59.30 FU MNTSTMT>MNTCAR1  Line= 109.008 F= 1 S=CaseDef  006
Msec=46770750  Cpu= 0 Mmem= 2052 Amem= 0

12.59.30 FU MNTSTMT>MNTCAR1  Line= 120.003 F= 1 S=CaseEnd  001
Msec=46770750  Cpu= 0 Mmem= 2052 Amem= 0

12.59.30 FU MNTSTMT>MNTCAR1  Line= 122.008 F= 1 S=CaseDef  006
Msec=46770750  Cpu= 0 Mmem= 2052 Amem= 0

12.59.30 FU MNTSTMT>MNTCAR1  Line= 101.008 F= 1 S=CaseDef  006
Msec=46770750  Cpu= 0 Mmem= 2052 Amem= 0
Maintain Statement Trace

Example: MNTCON DEBUG EX MNTCAR1 Using MNTSTMT

Note lines for Maintain statements Reposition, Stack Clear, Next and For statements:
Enabling Server Tracing and Setting Trace Levels

Server tracing must be enabled in order to activate the Maintain statement trace. These trace settings can be set:

- In the EDASPROF.PRF or user profiles.
- Using SYS_MGR.FOCSET to control tracing from within a specific line in a Maintain procedure.
  For more information on this command, see the Maintain Language Reference manual.

Refer to the WebFOCUS documentation for your server platform for information on setting traces from the Web Console and for profile commands.

Reference: Examples of Enabling and Setting Tracing

The following are examples of enabling server tracing in EDASPROF.PRF and setting tracing using SYS_MGR.FOCSET.

Example: Enabling Server Tracing in EDASPROF.PRF

The following commands enable server tracing and disable all trace levels except MNTSTMT:

```plaintext
SET TRACEUSER=ON
SET TRACEOFF=ALL
SET TRACEON=MNTSTMT
```

Example: Setting Tracing Using SYS_MGR.FOCSET

The following commands enable tracing from a Maintain procedure:

```plaintext
SYS_MGR.FOCSET("TRACEON", "MNTSTMT");
SYS_MGR.FOCSET("TRACEUSER", "ON");
```
Viewing Trace Output

After compiling and enabling the trace facility, running the Maintain procedure using the appropriate command (MNTCON RUN, MNTCON RUNIMAGE, MNTCON EX or MNTCON DEBUG EX) will generate the trace information. The output will be in the .trc file (ts000001, etc.) in the edatemp directory; each line of the Maintain trace information is prefixed with 'MNTSTMT'.

You can view the trace output file from the console by clicking Show the File. This displays the Maintain statement trace output for the .trc file displayed in the Select File dropdown box.

You can also generate custom reports using the MNTSTMT.MAS file, which is packaged with the server; the fields in this master file correspond to the components of the Maintain trace line. By issuing a FILEDEF for the trace output file, you can extract the relevant information.

Example: Viewing Trace Output in the MNTSTMT.MAS File

The following code

```plaintext
FILEDEF MNTSTMT DISK E:\IBI\APPS\MY71APP\TS000001.TRC
-RUN
TABLE FILE MNTSTMT
PRINT CPU MNTMEM AGNMEM
BY TOD BY PROCNAME BY STMTTYPE
IF MNTSTMTLAB EQ 'MNTSTMT>'
ON STMTTYPE SKIP-LINE
END
```

produces output formatted as shown below:

<table>
<thead>
<tr>
<th>TOD</th>
<th>PROCNAME</th>
<th>STMTTYPE</th>
<th>CPUMSEC</th>
<th>MNTMEM</th>
<th>AGNMEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.28.08</td>
<td>MNTNEW1</td>
<td>CaseDef</td>
<td>11010</td>
<td>2315</td>
<td>11016</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CaseEnd</td>
<td>11010</td>
<td>2316</td>
<td>11016</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MntForSt</td>
<td>11010</td>
<td>2316</td>
<td>11016</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NextStmt</td>
<td>11010</td>
<td>2316</td>
<td>11016</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11010</td>
<td>2316</td>
<td>11016</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11010</td>
<td>2316</td>
<td>11016</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11010</td>
<td>2316</td>
<td>11016</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RepoStmt</td>
<td>11010</td>
<td>2316</td>
<td>11016</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StakClear</td>
<td>11010</td>
<td>2316</td>
<td>11016</td>
</tr>
</tbody>
</table>
Sample Usage Scenarios for Tracing

- **Problem:** During a system/unit test, the programmer finds the application is failing in a case block and wants to know the specific PERFORM statement that triggered the CASE.
  
  **Solution:** Recompile with the DEBUG option and run application with tracing turned on. Inspect the trace file.

- **Problem:** A completed application is performing poorly but the source of the poor performance is not clear.
  
  **Solution:** SET TRACEON=MNTSTMT and SET TRACEON=MNTPERF without recompiling for debugging. Run the application with tracing on and use the trace analysis FEXs to pinpoint the case causing the problem. Follow standard debugging procedures.

- **Problem:** Same as above, but the application is in production and the test system does not exhibit problems.
  
  **Solution:** Use a user profile and SET TRACEUSER=ON for just one user so you can trace without affecting all users.

  After the new Maintain statement trace options have isolated the problem area, other trace options may be used to help pinpoint the source.

- **Problem:** The above analysis has shown that the problem resides in SQL database access.
  
  **Solution:** Add SET TRACEON=STMTRACE so you can see the Maintain statement trace with the SQL used against the database engine.

**MNTCON PERFORMANCE_ANALYSIS**

Use the MNTCON PERFORMANCE_ANALYSIS ON command from within a server global or user profile to generate timestamps for Maintain CALLs and EXECs without requiring other traces. This can help you pinpoint which procedures are most heavily called within an application. The output goes to the EDAPRINT.LOG file, and you can easily extract the format to FOCUS reports for analysis.

For more information, see the *Maintain Language Reference* manual.

**Additional Trace Settings**

Two additional server traces that are often useful when debugging Maintain applications are MNTDBMS and STMTRACE.

- **SET TRACEON=MNTDBMS** provides information on DBMS database activity by displaying the setup for calls to DB I/O services.
SET TRACEON=STMTRACE/1 and STMTRACE/2 trace SQL syntax by displaying SQL syntax traces (/1 statement, /2 substatement).

MTDBMS can be found under the Maintain Programmer Traces heading; STMTRACE is available under the Command Processing heading.
Deploying Legacy Applications to the Web

In early versions of WebFOCUS Maintain 4 and WebFOCUS Maintain 3 (known as Cactus), you could deploy applications either to the web or to Windows (also referred to as Client/Server).

When you deploy an application to the web, it resides on a central WebFOCUS server and is downloaded to an end user’s computer at run time. When you deployed an application to Windows, WebFOCUS Maintain created a stand-alone run-time application that needed to be installed on end user computers.

**Note:** WebFOCUS Maintain no longer supports Windows-deployed applications.

---

**In this appendix:**
- What Are the Differences Between Windows-Deployed and Web-Deployed Applications?
- Legacy Form Properties and Behavior
- Upgrading Legacy Windows Applications

---

**What Are the Differences Between Windows-Deployed and Web-Deployed Applications?**

The major difference between web and Windows deployment is the behavior of forms. The back-end logic (the commands that read data from and write data to the data sources) is the same in both types of deployment.

**Legacy Form Properties and Behavior**

The following form properties were applicable to Windows-deployed applications only and are no longer available because in web applications, a form is regulated by the browser window in which it is displayed:

- Height
- Left
- MaximizeBox
- MinimizeBox
Windows applications could have multiple forms visible to the end user (only one was the active form), while web applications only display one form at a time.

Upgrading Legacy Windows Applications

To upgrade your legacy Windows-deployed application:

1. Migrate your application to Version 7.
2. Deploy your application locally and test it.
This glossary explains key concepts related to the Maintain Suite of documentation.

<p>| <strong>Access file</strong> | Contains information needed by WebFOCUS to access the data in a data source. The Access File includes the real name and location of the data source. |
| <strong>application</strong> | A group of procedures and other components that work together to achieve a common goal. An application can be distributed across many servers, access varied data sources, and support multiple users. |
| <strong>Boolean expression</strong> | An expression in a programming language that returns a value of true (1) or false (0), based on the outcome of two or more relational expressions. Boolean expressions are often used in conditional expressions. |
| <strong>class</strong> | The template of an object. A class defines what type of object it is, and what it can do using variables and functions. |
| <strong>Controls Palette</strong> | A component in the Form Editor that contains the controls you can place on a Maintain form, such as text, radio buttons, check boxes and so on. |
| <strong>data access logic</strong> | The code that extracts data from the data source, manipulates it, and writes it back to the data source. All data access logic is coded using Maintain language. |
| <strong>data source stack</strong> | A non-persistent (or in-memory) table where you can store and manipulate data from one or more data sources. WebFOCUS Maintain procedures use stacks to hold values read from the data source and to manipulate data before writing it back to the data source. |
| <strong>database logic</strong> | The code that implements NEXT, INCLUDE, UPDATE, DELETE, and REVISE commands. |</p>
<table>
<thead>
<tr>
<th><strong>DBA (Database Administrator)</strong></th>
<th>A user, whose role is to perform development, design, and monitoring of the database. A DBA has unlimited access to the Master File and data source, and can set up or change security restrictions for individual users.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>deep copy</strong></td>
<td>Copies both the appearance of a control and any underlying data bindings, event handlers, and connections to resources. See shallow copy.</td>
</tr>
<tr>
<td><strong>deployment scenario</strong></td>
<td>Maps all of the components in your project to a WebFOCUS Server or a web server.</td>
</tr>
<tr>
<td><strong>event</strong></td>
<td>An action that an end user performs, such as opening a form, clicking a button, or moving to a field.</td>
</tr>
<tr>
<td><strong>event handler</strong></td>
<td>A programmable response to an action performed by a user, such as entering text in a field or clicking a button.</td>
</tr>
<tr>
<td><strong>form</strong></td>
<td>An interface object that enables you to interact with an application. You can create a Maintain application containing multiple forms.</td>
</tr>
<tr>
<td><strong>Form Editor</strong></td>
<td>A tool in the Maintain Development Environment (MDE) that enables you to design the user interface for your Maintain application.</td>
</tr>
<tr>
<td><strong>front end</strong></td>
<td>The user interface with which the end user interacts. This generally includes forms, web links, and scripts.</td>
</tr>
<tr>
<td><strong>function</strong></td>
<td>A series of commands in a procedure grouped together as a unit of control. Also known as cases.</td>
</tr>
<tr>
<td><strong>import module</strong></td>
<td>A non-executable procedure that contains Maintain language code.</td>
</tr>
<tr>
<td><strong>Java applets</strong></td>
<td>Compiled Java programs that can perform interactive animation, immediate calculations, or other simple tasks without having to send a user request back to the server.</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>JavaScript</strong></td>
<td>A small, object-based scripting language that you can use to create functions that can be embedded in or called from HTML pages. When you execute your application, your browser will interpret the code and perform the actions required.</td>
</tr>
<tr>
<td><strong>logical transaction</strong></td>
<td>A collection of data source operations that can access several data sources and that can be issued from several procedures.</td>
</tr>
<tr>
<td><strong>Master File</strong></td>
<td>A file that describes the data used by WebFOCUS Maintain. The extensions are .mas and .acx.</td>
</tr>
<tr>
<td><strong>member functions</strong></td>
<td>A class property that defines what you can do to objects of a class, and how you can manipulate the data. They exist only as members of the class.</td>
</tr>
<tr>
<td><strong>member variables</strong></td>
<td>A class property that defines data in the form of variables, existing only in the class.</td>
</tr>
<tr>
<td><strong>N-Tier Applications</strong></td>
<td>Applications capable of distributing processing over multiple platforms, which allows the creation of relationships among disparate data sources and increased application speed.</td>
</tr>
<tr>
<td><strong>numeric expression</strong></td>
<td>An expression in a programming language that performs a calculation using numeric constants, fields, operators, and functions.</td>
</tr>
<tr>
<td><strong>object</strong></td>
<td>An instance of a class that encapsulates both data and processes.</td>
</tr>
</tbody>
</table>
### Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output window</strong></td>
<td>An area in the Maintain Development Environment (MDE) that displays messages generated by WebFOCUS Maintain as well as Search Project results.</td>
</tr>
<tr>
<td><strong>persistence logic</strong></td>
<td>When an application processes a Winform_Show command, a formatted HTML screen is returned to the user, and the procedure and processing agent &quot;waits,&quot; or persists, until the user invokes another submit through the Servlet back to the running procedure. The Attach Manager of the server, using the cookie, reattaches the user to the waiting procedure and processing agent, and the processing continues. In this way, the waiting procedure and processing agent retain the context of the user between the different forms or screens of the application user experience.</td>
</tr>
<tr>
<td><strong>presentation logic</strong></td>
<td>Maintain syntax and browser script code that describes and implements the user interface for the application.</td>
</tr>
<tr>
<td><strong>pre-database stack processing</strong></td>
<td>Data source stacks that must be fully transformed to match the data control being used on the form before being returned to presentation logic procedure.</td>
</tr>
<tr>
<td><strong>property sheet</strong></td>
<td>Enables you to view and edit properties and events for your forms and controls.</td>
</tr>
<tr>
<td><strong>Resource Wizard</strong></td>
<td>A tool that helps you add resources, such as image files, HTML files, Java applets, and ActiveX controls to your project. Resources are non-native Maintain project components that you can use in the forms you design.</td>
</tr>
<tr>
<td><strong>script</strong></td>
<td>A component in the WebFOCUS Maintain project that contains stand-alone functions, written in either JavaScript or VBScript for execution on the client.</td>
</tr>
<tr>
<td><strong>set-based processing</strong></td>
<td>Enables you to read, manipulate, and write groups of records at a time. The data is manipulated using a data source stack.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>shallow copy</td>
<td>Copies the appearance of the control. See deep copy.</td>
</tr>
<tr>
<td>starting procedure</td>
<td>A procedure that you designate to be executed first, when running a project from the development environment.</td>
</tr>
<tr>
<td>Synonym Editor</td>
<td>A tool that enables you to view and edit the attributes of a synonym.</td>
</tr>
<tr>
<td>transaction integrity</td>
<td>A condition where the transaction is written to the data source only if all of its component commands are successfully written to the data source. If one part of the transaction fails, Maintain automatically rolls back the entire transaction, leaving the data source unchanged. Transaction integrity also ensures that when several users share access to the same data source, concurrent transactions run as if they were isolated from each other.</td>
</tr>
<tr>
<td>user context</td>
<td>The value of the variables and stacks of an application, the position in the database, and the state of the transaction at any given time.</td>
</tr>
<tr>
<td>variable</td>
<td>A named storage location capable of containing a certain type of data that can be modified during program execution.</td>
</tr>
<tr>
<td>VBScrip</td>
<td>A scripting language similar to JavaScript, except that it is only supported in Microsoft Internet Explorer.</td>
</tr>
<tr>
<td>web link</td>
<td>An object that contains a valid URL, such as a web page, email address, or FTP server.</td>
</tr>
</tbody>
</table>
Index

.fcm file 122, 345
.mnt file 122
.trc file 392
.wfm file 122

(GroupCode) property 267
(Name) property 267

3GL programs 32, 315

A

A0 variables 79, 182
  multi-edit boxes and 182
Access Files 43
active form 116
active layer in Form Editor 105
ActiveX controls 253, 254, 257, 258
  changing properties 257
  placing in forms 254
  unstable 258
Add Parameters command 304
AddFont function 206, 209
adding bookmarks 42
adding directories to project paths 34
adding existing files to project 31
AddTab function 206
Alert function 136
aligning controls 98
aligning text in a control 267
Alignment property 267
Alphanumeric format 78, 79
Alt property 268

AlternateRowColor property 268
amper variables in WebFOCUS reports 157, 237
  specifying in URL Wizard 237
applications 28, 121, 122, 123, 125, 127, 297, 348, 350, 351, 358, 383
  closing 350
  closing at run time 121, 125, 127
  debugging 383
  driver procedures 122, 123
  exiting at run time 121
  logic 358
  partitioning 297, 351
  security 348
arguments
  for functions 60
  for Maintain procedures 301
arguments for functions 61
asterisks next to project name 33
AT keyword 300
AUTO_COMMIT command 376

B

BackColor property 269
BackColorOver property 270
background images 223
BackgroundColor property 270
backslash character
  in image paths 223
Binding the Selection Result dialog box 179
blue properties in property sheet 89
Blur event 129, 199
  in grids 199
bookmarks
  deleting 42
  in text editor 42
Border property 271
BorderColor property 272
BorderText property 272
BorderWidth property 273
bottom alignment of controls 99
Bottom property 273
Bracket negative edit option 78, 80
broadcast commit 364
button control 165
changing properties 165
placing on forms 165

C

C/C++ programs 315
Calendar property 273
CALL command 131, 297, 298, 300, 301, 302, 314, 353
  in Event Handler editor 131
  KEEPing or DROPping the server session 301
  modularizing code 297
  passing parameters 301, 302
  specifying a server 300
  syntax 298
called procedures 302, 303
  FROM and INTO parameters 303
  input parameters 302
  output parameters 302
Cascading Style Sheets (CSS) 113
case sensitivity 143, 274
  in edit boxes 274
  in scripts 143
cases in functions 328
CaseStyle property 274
CellGetReadOnly function 206
CellGetTextColor function 206
CellSet function 212
CellSetText function 211
CellSetTextColor function 211
Change event 129, 175, 248
change verification strategies 369, 376, 379, 380
  for DB2 data sources 376, 379, 380
  for FOCUS data sources 369
changing project paths 34
check box control 167, 168
  placing on forms 167
  setting values 168
  triggering actions 168
Check event 129, 168
Checked keyword 168
Checked property 169, 275
CICS transactions 32, 315
Class Editor 330, 336, 337, 340
  Description tab 330, 340
  Functions tab 330, 337
classes 58, 78, 81, 327, 328, 330, 331, 332, 333, 334, 335, 340
  applying 78, 81
  defining 327, 330, 331, 332, 333
  deleting 330, 335
  DESCRIBE command and 330, 333
  displaying in a folder 58
  editing 330, 334
  inheritance 330
  libraries 340
  member functions 328
  member variables 328
  renaming 330, 335
  subclasses and superclasses 330, 332, 333
Clear All Bookmarks command 43
Click event 129
ClickArea event 129
ClickColumn parameter 141, 204, 205
ClickLink event 129, 204, 205
ClickMenuItem event 129
ClickRow parameter 141, 204, 205
clip setting for Overflow property 288
clipboard 94
Close event 129
CLOSE keyword in Winform command 117, 118
Close_A 117
Close_All keyword in Winform command 119
closing a form at run time 125, 127
closing application at run time 125, 127
closing WebFOCUS Maintain applications 350
COBOL programs 32, 315
color properties 260, 262, 268
  setting dynamically 262
coloring alternate rows in HTML tables 268
colors 211, 260, 261
  applying in Form Editor 260, 261
  customizing in Form Editor 261
  setting in grid with CellSetTextColor function 211
  setting in grid with QuickSetTextColor function 211
columns of report 318
Columns property 250, 275
combo box control 172, 173, 174, 175, 181
  changing properties 181
  defining list 173
  placing on forms 172
  setting values dynamically 174
  triggering actions 175
Comma inclusion edit option 78, 80
COMMIT command 361, 376
  DB2 data sources 376
  defining a logical transaction 361
COMMIT parameter 368
communication configuration files 373, 375
  for WebFOCUS Servers 373, 375
COMPILE command 389
compiling Maintain procedures 345, 389
Computed Stack Column dialog box 67, 69, 71
concurrent processing 366, 369
configuration files for WebFOCUS Servers 373, 375
  communication configuration files 373, 375
  EDACSG 373, 375
  odin.cfg 373, 375
confirm function 136
Content property 275
Control Columns dialog box 195
controls 85, 88, 94, 95, 96, 97, 98, 100, 101, 102, 103, 163, 164, 259, 264, 267
  adding to forms 88
  aligning 98
  center alignment 100
  changing order of 102
  changing tab order 264
  copying 95
  cutting 95
  duplicating 96
  grouping 101
  layering 103
  manipulating at design time 85
  manipulating at run time 259
  moving between layers 103
  pasting 96
  properties 267
  regrouping 102
  resizing 97
  selecting 94
  spacing 101
Controls palette 87, 88
converting input to uppercase or lowercase 274
Copy command 40, 95
  copying functions to procedures 63
  copying text 40
Create New Stack dialog box 72, 78
creating layers in Form Editor 104
creating projects 28, 29
  components 29
Index

creating script libraries 29
Credit negative edit option 78, 80
CSS support 113
CSSClass property 276
CSSName property 275
currency 78, 80
formatting 78, 80
CursorPointer property 276
customizing parser activity 39
Cut command 40, 95

data continuity 313
data source
editing descriptions 43
viewing structures 45
data source descriptions 29, 35, 36, 43, 45
creating 29
deleting 36
removing from project 36
renaming 35
viewing structure in Project Explorer 45
data source stacks 55, 58, 72, 73, 74, 75, 76, 187, 189, 193
creating 72, 73
creating from Select Segment Fields dialog box 187, 189
displaying implied columns 72
displaying in a folder 55, 58
editing 72, 74, 75, 76
viewing 193
data sources 29, 31, 43, 58, 59, 311, 359, 362, 366, 369, 370
accessing in child procedures 311
adding to projects 31
command failure 362
FOCUS Database Server 369
logical transactions 359
position in 311
position within logical transactions 362
data sources (continued)
reading with report procedures 370
sharing 366
specifying in a procedure 58, 59
data types 67, 78, 302, 312, 327, 328, 330, 331, 333
as part of variable definition 67
classes 327, 330, 331, 333
matching in procedure parameters 302, 312
database logic 351, 353
date formats 78, 79
DB2 data sources 376, 377, 379, 380
change verification 376, 379, 380
data adapter differences 376
transaction processing 376, 377, 379, 380
DBA 45, 348
specifying password 45
supplying information at run time 348
DBAPASS 348
DEBUG command 389, 390
debug option for deploying Maintain applications 389
debugging WebFOCUS Maintain applications 383
Decimal format 78
Declaration tab in Variable Editor 67, 69
DESCRIBE command 330, 333
DECLAIM command 323, 341, 342
variable binding 323
deep copy 94, 96
DefaultButton property 277
defining project flow 32
Definition tab 72, 75
DELETE Maintain command 353
deleting projects or project components 36
deploying applications 373
DESCRIBE command 330, 333
Description tab 57, 64, 71, 76, 154, 330, 340
in Class Editor 330, 340
in Function Editor 64
in Link Editor 154
Description tab (continued)
in Stack Editor 76
in Variable Editor 71
developing applications in teams 54
Display all files in the project paths button 35
displaying information about an image 268
Double Precision format 78, 80
DoubleClick event 129
adding data source fields to a form 189
adding stack columns to grids or HTML Tables 194
copying functions to a procedure 63
copying text 40
copying variables to a procedure 69
creating data source stacks 73
defining lists in the List Source dialog box 173
duplicating controls 96
der dynamically setting color properties 262
generating code to display a form 114, 115
generating Winform Set syntax 119, 120, 259
inserting IWCTrigger into a script 137
moving functions to another procedure 62
moving text 40
moving variables to another procedure 69
placing ActiveX controls on forms 254
placing images on forms 223
placing Java applets on forms 233
setting properties dynamically 89, 91
specifying functions as event handlers 131, 132
using an import module in a procedure 82
drill down feature 322
drilling down in HTML Tables 204
driver procedures 122, 123
DROP keyword 301
Duplicate command 96

E

EDACSG ddname 373, 375
EDAPRINT.LOG filelog file 387, 388
EDASPROF global server profile 372
FOCUS Database Server 372
edit boxes 182, 185, 274
changing properties 185
converting input case in 274
placing on forms 182
Edit Class dialog box 330, 336, 337, 338
Edit event handlers command/button 126
Edit Function dialog box 63, 64
Edit menu 38, 39, 40, 41, 45, 85, 94, 95, 96, 103
Copy 40, 95
Cut 40, 95
Duplicate 96
Find 41
Password 45
Paste 40, 96
Paste appearance 96
Redo 40, 103
Replace 41
Search project 38
Select all 40, 94
Undo 40, 103
Edit parameters command 304, 305
Edit Variable dialog box 67, 69
editing FOCUS procedures 316
Editor toolbar
Clear All Bookmarks 43
Find 41
Next Bookmark 43
Previous Bookmark 43
Toggle Bookmark 42
e-mail address 143
Email Address dialog box 161
e-mail links 143, 148, 160
e-mail messages 162
Enabled property 278

END keyword in GOTO command 363

document user input 135, 138
    using scripts 135, 138
    validating 135

Enter a List Item dialog box 172, 179

Enter a Parameter dialog box 158, 233, 237

environment variables 373, 375
    HOME 373, 375

error messages 36

Event Handler editor 38, 125, 126, 127

events 125, 126, 129, 131, 199
    common combinations 131
    read/write grids and 199
EXEC command 131, 297, 298, 300, 301, 312, 314, 355
    in Event Handler Editor 131
KEEPing or DROPping the server session 301
    modularizing code 297
    passing parameters 312
    specifying a server 300
    syntax 298
EXEC logic 351, 355

executing Maintain procedures 346

Explorer 33, 35
    viewing new file types 35

external procedures 32, 315, 373, 374
    deployment considerations 373
    reporting 373

F

fcm extension 345

FDS (FOCUS Database Server) 368, 369, 370, 371, 372, 373
    change verification 369
    deployment considerations 373
    identifying 372
    report procedures 373
    SET PATHCHECK 371
    transaction processing 370

File menu 42

files 29, 31, 35, 36
    adding to projects 31
    creating 29
    deleting 36
    removing from project 36
    renaming 35
    viewing in project path 35
    viewing types 35

Find command 41, 43
    using with bookmarks 43

FixedColumns property 278

Floating dollar edit option 78, 80

Floating Point format 78, 80

flow of control 114

FOCCOMP file type or ddname 345

FocCurrent variable 365, 369
    change-verify protocol 369

FocIndex 205

FOCSET command 391

FOCUS code 315

FOCUS data sources 368, 369, 370, 371, 372
    change verification 370
    concurrent transactions 368
    FOCUS Database Server 369
    SET COMMIT 368, 369
    SET PATHCHECK parameter 371
    sharing access 369
    transaction processing 368, 371

FOCUS Database Server (FDS) 368, 369, 370, 371, 372, 373
    change verification 370
    deployment considerations 373
    identifying 372
    report procedures 373
    SET PATHCHECK 371
    transaction processing 369

Focus event 129, 199
    in grids 199

folders for procedure components 55, 58
Font property 279
fonts 209
  changing in read/write grids 209
ForeColor property 279
ForeColorOver property 280
Form Editor 85, 86, 87, 92, 93, 103, 105, 106, 260, 261
  applying colors 260, 261
deleting layers 106
drawing aids 92
grids 92
guidelines 87, 92
layout 87
locking layers 105
rulers 87, 93
undoing and redoing actions 103
unlocking layers 105
Form menu 96, 103
  Duplicate 96
  Edit layers 103
Format data type 78
forms 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 122, 123, 125, 127, 142, 143, 267
  active 116
  changing properties 112
closing at run time 117, 118, 119, 125, 127
creating 109, 110
deleting 110
displaying at run time 114, 115, 116
driver procedures 122, 123
editing 110, 111
flow of control 114
manipulating at run time 114
navigating 110
non-persistent 116
persistent 116
properties 267
renaming 110, 111
style sheets 113
using script libraries 142, 143
frame control 191, 192
  changing properties 192
  placing on forms 191
FROM keyword 301, 302, 312
  in CALL command 301, 302
  in EXEC command 312
FTP servers 143, 147
defining a Web link 147
Function Parameter dialog box 69, 71
Function Return dialog box 67, 69, 71
functions 131, 132, 133, 134, 135, 143, 328
  JavaScript and VBScript 132, 133, 134, 135, 143
  member 328
  running from Event Handler editor 131, 132
Functions tab in Class Editor 330, 337

G

General tab 283, 330
  Class Editor 330
General toolbar
  Copy 40, 95
  Cut 40, 95
  Paste 40, 96
  Print 42
  Redo 40, 103
  Save 34
  Save all 34
  Search project 38
  Undo 40, 103
Get keyword in Winform command 119, 120, 259
GetCell function 209, 211, 212
GetCurrentColumn function 206, 209
GetCurrentRow function 206, 209
GetGridDefault function 206
GetNumberColumns function 211
GetNumberRows function 211
gif images 223
GOTO command 313
GOTO END command 363, 364
GoToCell function 206, 210
GoToColumn function 206, 210

graphics 32, 223

Grid Column Properties dialog box 197

Grid command 92

grid control 193, 194, 199, 201, 205, 206
changing properties 201
functions 205, 206
placing on forms 194

Grid settings command 92

Grid Settings dialog box 92, 93

GridLines property 280

grids 92
in Form Editor 92
group box controls 217
changing properties 217
placing on forms 217

Group command 101

GroupCode property 267

grouping controls 101

guidelines 92

HTML Content Source dialog box 218, 220

HTML File Editor 38

HTML files 29, 31, 35, 36, 143
adding to projects 31
creating 29
deleting 31
removing from project 36
renaming 35

HTML Object control 218, 219, 222, 317
changing properties 222
displaying WebFOCUS reports 317
placing on forms 218, 219

HTML Table control 193, 194, 201, 204, 205, 318
changing properties 201
creating links 204
determining cell values 205
determining columns 205
determining rows 205
displaying report results 318
placing on forms 194

HTMTABLE format for WebFOCUS reports 317, 318

HTTP Server 147

Hyperlink property 283

IBISTART.WFS file 348

IBIValidation property 283

image control 223, 228, 232, 233
changing at run time 232, 233
changing properties 228
placing on forms 223

Image Map dialog box 229, 231

Image property 285

Image Source dialog box 223, 225

ImageDown property 232, 285

ImageOver property 232, 285

images 223, 224, 227, 229, 231
adding to projects 224

HOME environment variable 373, 375
images (continued)
  as controls 223
  as form background 223
  maps 229, 231
implied columns in a data source stack 72, 73
IMPORT keyword in MODULE command 341
import modules 29, 34, 35, 36, 81, 82, 340
  creating 29
  deleting 36
  in procedures 82
  removing from projects 36
  renaming 35
  saving 34
  using as class libraries 340
Import these Modules dialog box 83
Import WebFOCUS Parameters command 323
Import WebFOCUS Parameters into Maintain dialog box 324
IMS/TM transactions 32, 315
inheritance 330
Initialize tab in Variable Editor 67, 71
input parameters
  specifying for called procedures 302
  specifying for calling procedures 302
input parameters for a procedure 301, 304
  specifying for called procedure 304
Insert ActiveX Control dialog box 256
Insert ActiveX Control Resource dialog box 257
INSERT Maintain command 353
Insert Text dialog box 182, 184
Integer format 78, 80
INTO keyword 301, 302, 312
  in CALL command 301, 302
  in EXEC command 312
ItemBorder property 285
IWCTrigger function 137, 255, 256
  passing ActiveX control values 255, 256
J
Java applet control 233, 235, 238
  changing parameters at run time 235
  changing properties 238
  placing on forms 233
Java Applet Parameters dialog box 236
Java applets 233, 234, 237
  adding as resources 234
  adding to projects 234
  parameters 237
JavaScript functions 132, 133, 134, 135, 140, 143,
  255, 256
  calling 133, 134, 135, 140
  debugging 143
  passing ActiveX control values 255, 256
  using for validation 135
  using in applications 133
  writing in Event Handler editor 133, 143
JavaScript libraries 29, 35, 36, 142, 143
  associating with forms 142, 143
  creating 29, 142
  deleting 36
  editing 142, 143
  removing from projects 36
  renaming 35
jpeg images 223
jumping to bookmarks in text editor 43
K
KEEP keyword 301, 313, 363
  in CALL command 301
  in EXEC command 301
  in GOTO command 313
KeyPress event 129
keyword in Winform command 117
L
labeling border of radio button 272
Index

Language Wizard 57
Layer property 285
layers 103
Layers property 106
Layers Sheet dialog box 103
Layout menu 87, 92, 97, 98, 99, 100, 101, 102
  Align (Bottom) 99
  Align (Center horizontally) 100
  Align (Center vertically) 100
  Align (Left) 98
  Align (Right) 99
  Align (Top) 99
Bring to front 102
Grid settings 92
Group 101
  Make same size 97
  Make same size (Height) 97
  Make same size (Width) 97
Send to back 102
Space 101
Ungroup 101
Layout toolbar 87, 92, 93, 97, 98, 99, 100, 101, 103, 126
  Align Edges (Bottom) 99
  Align Edges (Center horizontally) 100
  Align Edges (Center vertically) 100
  Align Edges (Left) 98
  Align Edges (Right) 99
  Align Edges (Top) 99
Edit event handlers 126
Edit layers 103
Group controls 101
Make Same Size 97
Space 101
Toggle grid 92
Toggle rulers 93
Leading zeros edit option 78, 80
left alignment of controls 98
line breaks in HTML Table/grid header titles 197
lines 239
  changing properties 239
  placing on forms 239
Link Editor dialog box 153, 154
links 143, 145, 148, 160, 204
  email 148, 160
  HTML tables 204
  web 145
list box control 172, 173, 174, 175, 181
  changing properties 181
  defining list 173
  placing on forms 172
  setting values dynamically 174
  triggering actions 175
List Source dialog box 172, 176
ListItems property 286
ListItems stack 174, 248
LockColumns function 206
LockRows function 206
logic in applications 358
logical transactions 359, 360, 361, 362, 363, 364, 365, 366, 367, 369, 373
  broadcast commit 364
  concurrent transactions 366, 367
  concurrent transactions in FOCUS Database Server 369
  data source position 362
  DBMS types 364
  defining 361
  deployment considerations 373
  ending an application 365
  failure 362
  FocCurrent 365, 366
  multiple data source types 364
  multiple servers 365
  open transactions when an application ends 365
  processing 359, 360
  rolling back 362
  spanning procedures 363
  success 364

M
Maintain commands 353, 355
  CALL 353, 355
Maintain commands (continued)
DELETE 353
EXEC 355
INSERT 353
NEXT 353
REVISE 353
UPDATE 353

Maintain Development Environment 27, 28, 33
opening 28

Maintain functions 58, 60, 61, 62, 65
creating 61, 65
displaying in folders 58
editing 62

Maintain libraries 29, 35, 36, 81, 82, 340
creating 29
deleting 36
in procedures 82
removing from project 36
renaming 35

Maintain procedures 29, 31, 34, 35, 36, 55, 56, 58,
59, 82, 301, 305, 311, 322, 345, 346,
363, 372, 375, 389
accessing data sources 311
adding to projects 31
compiling 345, 389
creating 29
data source position 311
deleting 36
editing 56
executing 346
executing from WebFOCUS procedures 322
executing from WebFOCUS report procedures 322
MODIFY 375
passing parameters 301
removing from project 36
renaming 35
saving 34
selecting starting procedure 56
setting parameters 305
sharing data sources with WebFOCUS Maintain
375
specifying data sources 58, 59
transaction integrity 363
using import modules 82
viewing components in folders 58

Maintain statement trace 388
Map property 229, 230, 286
Mask function 207

Master File Editor 38
Master Files 29, 31, 35, 36, 43, 45
adding to projects 31
creating 29
deleting 36
removing from project 36
renaming 35
viewing structure in Project Explorer 45

Member Function dialog box 63, 64
member functions 328, 330
inheritance 330

Member Variable dialog box 67, 69, 71
member variables 328, 329, 330
inheritance 330

memory management 313
menu control 240, 245
adding 240
changing properties 245

Menu Items dialog box 244

miscellaneous business logic 351, 353, 354

MNTCON COMPILE command 345
MNTCON EX command 346
MNTCON RUN command 346
MNTPERF setting 388

MNTSTMT setting 388, 389, 390, 391
MNTSTMT.MAS file 392

MODIFY applications 375
modular source code 297

MODULE command 341

monetary values 78, 80

MouseDown event 129
MouseMove event 129
MouseOut event 129
MouseOver event 129
MouseUp event 129
moving functions to another procedure 62
moving text 40
multi-edit boxes 182, 185
changing properties 185
placing on forms 182
MultiSelection property 286

N
Name property 110, 111, 267
naming controls or forms 267
New Class dialog box 330, 336
New Function dialog box 63, 64
New Variable dialog box 69, 71
New Window Options dialog box 125, 151
Next Bookmark icon 43
NEXT Maintain command 353
numeric formats 78, 80

O
objects 327, 341, 342
declaring 341, 342
Variable Editor 341
odin.cfg file 373, 375
OnCanViewMove event 199
OnCBDClicked event 199
OnCBLClicked event 199
OnCBRClicked event 199
OnCellChange event 199
OnCellChanged event 199
OnCharDown event 199
OnColChange event 199
OnColSized event 199
OnEditFinish event 199
OnEditStart event 199
OnHitButton event 199
OnKeyDown event 199
OnLClicked event 199
OnLoad property 287
OnRCClicked event 199
OnRowChange event 199
OnRowSize event 199
OnRowSizing event 199
OnSHDClicked event 199, 212
OnSHLCicked event 199, 212
OnSHRClicked event 199, 212
OnTHDClicked event 199, 212
OnTHLClicked event 199, 212
OnTHRClicked event 199, 212
Open dialog box 52
Open event 129
Oracle procedures 32
order of layers in Form Editor 105
Orientation property 287
output parameters 302, 304, 306
specifying for called procedures 302, 304
specifying for calling procedures 302
output parameters for a procedure 301
Output window 36
Overflow property 288
parameters 148, 157, 301, 323
  adding 148
  binding 148
  passing from WebFOCUS procedures 323
  specifying in URL Wizard 157
Parameters dialog box 233, 236
parent target name 150
parser activity 39
partitioning code 297
PASS parameter 348
Password command 45
Password property 288
passwords 45, 288
  displaying asterisks instead of contents 288
  specifying for DBA 45
Paste appearance command 96
Paste command 40, 96
PATHCHECK parameter 371
PCHOLD files 317
PenWidth property 289
performance transaction processing 363
persistence management 356
pictures 223
PreSelectedItems stack 174
presentation logic 351, 352
Previous Bookmark icon 43
Print command 42
Print preview command 42
printing components as text 42
Procedure Editor 38, 302
procedure parameters 302
project components 29, 31, 34, 35, 36
  adding to projects 31
  creating 29
  deleting 36
  removing from project 36
  renaming 35
  saving 34
Project Explorer 33, 45
  displaying data source description structure 45
project paths 28, 34
  changing 34
  specifying at creation time 28
projects 28, 29, 32, 34, 35, 36, 37, 39, 56
  changing paths 28, 34
  components 29
  creating 28
  defining flow 32
  deleting 36
  opening components as text 39
  renaming 35
  saving 34
  searching text 37
  Starting Object 56
  viewing/hiding files in path 35
prompt function 137
prompted edit boxes 100, 187
  aligning 100
properties 89, 91, 119, 120, 259, 267
  changing 91
  changing dynamically 91
  determining at run time 119, 120, 259
  for forms and controls 267
  setting at run time 119, 120, 259
property sheet 87, 89
pull-down menus 240, 241
QuickSetAlignment function 206
QuickSetBackColor function 206
QuickSetFont function 206, 209
QuickSetMask function 207, 210
QuickSetText function 209
QuickSetTextColor function 206, 211

R
radio button control 246, 247, 248, 250
determining layout 250
placing on forms 247
setting values dynamically 248
triggering actions 248
RDBMS stored procedures 32, 315
read-only access to grid cells 209
read/write grids 193, 194, 199, 201, 205, 206
changing properties 201
events 199
functions 205, 206
placing on forms 194
ReadOnly property 289
Redo command 40, 103
redoing actions 103
RedrawAll function 206, 209, 210, 211
Regroup command 102
regrouping controls 101
removing layers from Form Editor 106
removing project components 36
renaming layers 106
renaming projects or project components 35
renaming WebFOCUS procedures 35
reordering layers in Form Editor 105
Replace command 41
replacing text 41
Report Painter 322
reports (continued)
specifying in URL Wizard 156
REPOSITION command 311
Reset keyword 119, 121
RESET keyword 313
resizing controls 97
Resource Wizard 46, 52, 144
specifying Active X controls 46
specifying parameters for applet 46
specifying paths 46
specifying unique name 52
return values for functions 60, 61
return values for Maintain procedures 301
REVISE Maintain command 353
right alignment of controls 99
Right property 289
ROLLBACK command 361, 362, 376
  DB2 data sources 376
Rows property 250, 290

S
Save all button/command 34
Save button/command 34
saving projects 34
  automatically 34
  components 34
Scientific notation edit option 78, 80
scope of variables 60
Script Editor 38
script functions 132, 133, 134, 135, 140, 143
debbuging 143
  for validation 135
  in applications 133
  running 133, 134, 135, 140
  writing in Event Handler editor 133, 134
script libraries 29, 35, 36, 142, 143
  associating with forms 142, 143
script libraries (continued)
  creating 142
  deleting 36
  editing 142, 143
  removing from projects 36
  renaming 35
scripts 132
scroll setting for Overflow property 288
Scrolling property 290
Search project command 38
searching for text in project 38
Secure Sockets Layer 142
security
  in WebFOCUS Maintain applications 348
Select all command 40, 94
select control 88
Select Segment Fields dialog box 187, 189
Select Stack Columns dialog box 187, 188
SelectedItem properties 291
SelectedItems Property 291
SelectedItems stack 175
selecting button as default 277
selecting check box as default 275
selecting text 40
self target name 150
self.WinClose function 117, 118, 127
self.WinExit function 117, 118, 121, 127
servers
  tracing 391
Set Check Box State dialog box 167, 169
Set keyword 120, 259
  in Winform command 259
SET parameters 368, 369, 371, 388, 391
  COMMIT in FOCUS data sources 368, 369
  PATHCHECK 371
  TRACEOFF 391
SET parameters (continued)
  TRACEON 388, 391
  TRACEUSER 391
SET PATHCHECK parameter 371
SetAlternateRowColor 268
SetBackColor 269
SetBackColorOver 270
SetBorderColor 272
SetCell function 209, 211
SetColumn function 206, 210
SetCurrentColumn function 206, 210
SetCurrentRow function 206, 210
SetForeColor 279
SetForeColorOver 280
SetHeaderBackColor 281
SetHeaderForeColor 282
SetTHNumberRows property 206
SetTHRowHeight property 206
setting selected value on form to variable or data
source stack columns 291
shallow copy 94, 96
shortcuts 39, 40, 62, 63, 67, 69, 72, 73, 82, 89, 91, 94, 96, 114, 115, 119, 120, 131, 132, 137, 173, 189, 194, 223, 233, 254, 259, 262
adding data source fields to a form 189
adding stack columns to grids or HTML Tables 194
copying functions to a procedure 63
copying text 40
copying variables to a procedure 69
creating data source stacks 72, 73
defining lists in the List Source dialog box 173
duplicating controls 94, 96
dynamically setting color properties 262
generating code to display a form 114, 115
generating Winform Set syntax 119, 120, 259
inserting IWCTrigger into a script 137
moving functions to another procedure 62
shortcuts (continued)
moving text 40
moving variables to another procedure 67, 69
placing ActiveX controls on forms 254
placing images on forms 223
placing Java applets on forms 233
setting properties dynamically 89, 91
specifying a function as an event handler 131, 132
using an import module in a procedure 82
Show All Files command 31, 35
Show keyword in Winform command 115
Show_Active keyword in Winform command 115
Show_And_Exit keyword 116
Show_Inactive keyword 116, 259, 262
color settings and 262
Winform Set and 259
SHOW_INACTIVE keyword
color settings and 260
Signature tab in Function Editor 63
signatures for functions 60
source management 54
Source property 291
spacing controls 101
specifying ability to adjust image size 291
specifying ability to scroll in frame 290
specifying Active X controls with Resource Wizard 46
specifying back color for header row of HTML table 281
specifying background color of form or control 269
specifying background color when cursor is on top of menu 270
specifying border for menu item 285
specifying border on controls 271
specifying bottom of control on form 273
specifying clickable parts of image 286
specifying color of control's border 272
specifying color of font for header row of HTML table 282
specifying columns and order of data source stack 275
specifying contents of frame 291
specifying control response to an event 278
specifying cursor type 276
specifying DBA passwords 45
specifying display of grid lines in grid or HTML table 280
specifying display of information for control or form 293
specifying display of menu bar horizontally or vertically 287
specifying font for header row of HTML table 281
specifying font for text in control 279
specifying header row of HTML table 282
specifying how a password displayed 288
specifying how much of HTML object or table appears 288
specifying HTML code inline 275
specifying image as background 270
specifying image that appears when mouse is over image 285
specifying image that appears when user clicks on image 285
specifying items listed in a group of controls 286
specifying layer that a control is on 285
specifying location of check box description 292
specifying location of right side of control 289
specifying multi-selection capability for list box 286
specifying name of image on form 285
specifying number of buttons in a row for radio button 290
specifying parameters for applet with Resource Wizard 46
specifying paths with Resource Wizard 46
specifying read-only feature of displayed value 289
specifying stationary columns in a grid 278
specifying text control as hyperlink 283
specifying text in a control 292
specifying the width of a line 289
specifying title on form's title bar 293
specifying unique name with Resource Wizard 52
specifying user's ability to tab to a control 291
specifying visibility or invisibility of control 294
specifying Web link for control or form 283
specifying width of control's border 273
Stack Editor 75, 76
Stack Overflow error 122
stacks 58, 72, 73, 74, 193
creating 73
displaying in a folder 58
editing 72, 74
viewing 193
Starting Object for projects 56
starting procedure 56
stored procedures in Maintain applications 315
Stretched property 291
style sheets 113
submenus 240, 241
Sybase procedures 32
SYS_MGR.FOCSET command 391
system messages 36
system resources 355

T

Tab Order dialog box 264, 265
Table Column dialog box 197
Tabstop property 291
team development 54
text
  entering 39
  finding 41
text color
  specifying in control 279
  specifying when cursor is on top of menu 280
text control 251, 252, 253
  changing properties 252
  placing on forms 252
  using as hyperlinks 253
text editor
  wrapping text 39
text editors 38
text files 29, 35, 36
  creating 29
  deleting 36
  removing from project 36
  renaming 35
Text property 292
text searches 37
TextOnLeft property 292
Title property 293
Toggle Bookmark command 42
Toggle grid button 92
Toggle rulers button 93
ToolTipText property 293
top alignment of controls 99
Top function 340

top target name 150

TRACEON setting 391

TRACEUSER setting 391

tracing 388, 391, 392, 393

viewing output 392

transaction integrity 359, 360, 361, 363, 365, 366, 370, 373, 376

across procedures 363

change verification 370

concurrent transactions 366

deployment considerations 373

FocCurrent 365

multiple servers 365

transaction locking strategies 376, 377, 379

transaction processing 359, 363, 364, 366, 369, 370, 372

broadcast commit 364

change verification 370

concurrent transactions 369

multiple data source types 364

performance 363

USE command 372

trc file 392

TYPE ON EDAPRINT command 387, 388

Type Wizard 78

UPDATE Maintain command 353

URL Link dialog box 192

URL tab in Link Editor 153

URL Wizard 144, 148, 155, 157, 160, 162

entering an email message 162

including email links 160

specifying amper variables 157

specifying parameters 157

URLs (Universal Resource Locators) 153

USE command with FOCUS Database Server 372

V

validating data format at run time 283

variable binding 323

limitations 323

variable length alphanumeric format 78, 79, 182

using with multi-edit boxes 182

variables 58, 67, 69, 71, 301, 313, 328, 341, 342

copying to a procedure 69

creating 68

defining objects 341, 342

deleting 67

displaying in a folder 58

editing 68

initializing 67, 71

memory management 313

moving to another procedure 69

passing between procedures 301

Variables tab in Class Editor 330, 338

VBScript functions 132, 133, 134, 135, 143, 255, 256

debugging 143

passing ActiveX control values 255, 256

running 133, 134

using for validation 135

using in applications 133

writing in Event Handler editor 133, 134

VBScript libraries 29, 35, 36, 142, 143

associating with forms 142, 143

creating 29, 142

UnCheck event 129

Undo command 40, 103

undoing actions 103

Ungroup command 101

ungrouping controls 101

unhiding layers in Form Editor 105

Universal Resource Locators (URLs) 153

unstable ActiveX controls 258
VBScript libraries (continued)
  deleting 36
  editing 142, 143
  removing from projects 36
  renaming 35
View menu
  Grid 92
  Rulers 93
viewing components as text 42
viewing error messages 36
viewing file types in Explorer 35
viewing files in the project path 35
viewing system messages 36
viewing trace output 392
Visible property 142, 294
  setting with scripts 142

W

Web Link dialog box 150
web links 143, 145, 149, 150
  creating 145
  deleting 150
  editing 149
  using as event handlers 149
WebFOCUS Developer Studio 27, 352
WebFOCUS Maintain 27
WebFOCUS Procedure Editor 38
WebFOCUS procedures 29, 31, 35, 36, 312, 315, 316, 317
  adding to projects 31
  creating 29
  deleting 36
  editing 316
  executing from Maintain procedures 316
  incorporating into projects 316
  passing parameters 312
  removing from projects 36
  renaming 35
  using report output 317
WebFOCUS reports 148, 156, 157, 233, 237
  defining Web link 148, 233, 237
  parameters 157, 233, 237
  specifying in URL Wizard 156
WebFOCUS Servers 300, 365, 372, 373, 375
  communication configuration files 373, 375
  configuration directory 373, 375
  EDACSG 373, 375
  edaserve 373, 375
  FOCUS Database Server profiles 372
  logical transactions spanning servers 365
  odin.cfg 373, 375
  specifying in CALL and EXEC commands 300
WF_USER cookie 348
wfmstart.html launch page 347
WinClose function 117, 118, 125, 127
Windows clipboard 94
WinExit function 121, 125, 127
Winform command 114, 115, 116, 118, 119, 120, 121, 259
  Close 118
  Close_All 119
  flow of control 114
  Get 120, 259
  GET 259
  Reset 121
  Set 120, 259
  SET 259
  Show_Active 115
  Show_And_Exit 116
  Show_Inactive 116
WINFORM command
  SET 259
  wizards 46, 57, 78, 144, 155
    Language Wizard 57
    Resource Wizard 46, 144
    Type Wizard 78
    URL Wizard 144, 155

X

x=? initialize tab in Variable Editor 67, 71
Z

Zero suppression edit option 78, 80
Customer Connections

Where users partner with Information Builders to provide feedback and create quality documentation.

Partner With Us

Information Builders Technical Content Management team is comprised of many talented individuals who work together to design and deliver quality technical documentation products. Your feedback supports our ongoing efforts!

You can also join a usability study to get an early look at new documentation products and services. Your participation helps us create documentation products that meet your needs.

To send us feedback or make a customer connection, contact Michele Brady, Service Director, Technical Content Management at Michele_Brady@ibi.com.

Request Permission to Use Copyright Material

To request permission to repurpose copyrighted material, please contact Frances Gambino, Vice President, Technical Content Management at Frances_Gambino@ibi.com.

Visit the Information Builders Technical Content Library

To visit the Technical Content Library, go to http://documentation.informationbuilders.com/.