

# **WebFOCUS**

Developing WebFOCUS Maintain Applications  
Version 5 Release 2

EDA, EDA/SQL, FIDEL, FOCCALC, FOCUS, FOCUS Fusion, FOCUS Vision, Hospital-Trac, Information Builders, the Information Builders logo, Parlay, PC/FOCUS, SmartMart, SmartMode, SNAPPack, TableTalk, WALDO, Web390, WebFOCUS and WorldMART are registered trademarks, and iWay and iWay Software are trademarks of Information Builders, Inc.

Due to the nature of this material, this document refers to numerous hardware and software products by their trademarks. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 2003, by Information Builders, Inc. All rights reserved. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Printed in the U.S.A.

---

---

## Preface

This documentation describes how to develop and deploy WebFOCUS Maintain applications using the Maintain Development Environment. It is intended for application developers who are responsible for planning the enterprise's software environment and for operating WebFOCUS Maintain. This documentation is part of the WebFOCUS Maintain documentation set.

---

## How This Manual Is Organized

---

This manual includes the following chapters:

Chapter		Contents
<b>1</b>	<i>Working in the Maintain Development Environment</i>	Describes how to create, develop, and edit projects using the Maintain Development Environment.
<b>2</b>	<i>Developing Procedures</i>	Describes how to develop procedures including passing parameters between procedures and using functions, variables, and data source stacks in procedures.
<b>3</b>	<i>Using the Form Editor</i>	Describes how to use the Form Editor to create forms.
<b>4</b>	<i>Developing and Using Forms</i>	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.
<b>5</b>	<i>Defining Events and Event Handlers</i>	Introduces the different types of events, and describes how to use event handlers.
<b>6</b>	<i>Developing and Using Controls</i>	Introduces all of Maintain's controls, and explains how to use them to create dynamic forms.
<b>7</b>	<i>Developing Classes and Objects</i>	Explains how your project can benefit from using classes and objects, and describes how to define classes, declare objects, and use class libraries.
<b>8</b>	<i>Debugging WebFOCUS Maintain Applications</i>	Describes how to use the WebFOCUS Maintain debugging tools to debug your applications.

Chapter		Contents
9	<i>Developing an Application for a Shared Application Server</i>	Describes how to create an application that is appropriate for use with the Shared Application Server.
10	<i>Managing Team Development</i>	Describes how multiple users can collaboratively develop a WebFOCUS Maintain project.
11	<i>Ensuring Transaction Integrity</i>	Describes how you can ensure transaction integrity at the application and data source levels.
12	<i>Editing Project Components as Text</i>	Describes how you can edit procedures and other application components as text.
13	<i>Customizing the Maintain Development Environment</i>	Describes how to customize the Maintain Development Environment toolbars, status bar, Application Explorer, and editors.

## About Your Documentation

---

WebFOCUS Maintain documentation includes one printed manual, a context-sensitive help system, and three corresponding manuals that are provided as PDF files. The manuals provided are:

- *WebFOCUS Maintain Getting Started* provides an introduction to WebFOCUS Maintain. It includes an overview of how to develop applications and a discussion of WebFOCUS Maintain concepts. A step-by-step tutorial is provided to prepare you for your first steps with the Maintain Development Environment. (Print and PDF.)
- *Developing WebFOCUS Maintain Applications*, which is this manual. (PDF only.)
- *WebFOCUS Maintain Language Reference* describes the Maintain language, including basic language rules, a comprehensive reference of Maintain commands and expressions, form and control properties, and error messages. (PDF only.)

Since WebFOCUS Maintain is part of the WebFOCUS Developer Studio suite of tools, you will also need to have available your WebFOCUS Developer Studio documentation.

## Documentation Conventions

---

The following conventions apply throughout this manual:

Convention	Description
<b>THIS TYPEFACE</b> or <i>this typeface</i>	Denotes syntax that you must enter exactly as shown.
<i>this typeface</i>	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
<u>underscore</u>	Indicates a default setting.
<i>this typeface</i>	Represents a placeholder (or variable) in a text paragraph, a cross-reference, or an important term.
<b>this typeface</b>	Highlights a file name or command in a text paragraph that must be lowercase.
<i>this typeface</i>	Indicates a button, menu item, or dialog box option you can click or select.
Key + Key	Indicates keys that you must press simultaneously.
{ }	Indicates two or three choices; type one of them, not the braces.
[ ]	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.
	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
...	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis points (...).
.	Indicates that there are (or could be) intervening or additional commands.

## **Related Publications**

---

The WebFOCUS Developer Studio documentation set may also contain information that is useful to you.

Visit our World Wide Web site, <http://www.informationbuilders.com>, to view a current listing of our publications and to place an order. You can also contact the Publications Order Department at (800) 969-4636.

## **Customer Support**

---

Do you have questions about WebFOCUS Maintain?

Call Information Builders Customer Support Service (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 WebFOCUS Maintain questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code (xxxx.xx) when you call.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our World Wide Web site, <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.

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 most effectively, be ready to provide the following information when you call:

- Your six-digit site code (xxxx.xx).
- Your WebFOCUS configuration:
  - The front-end you are using, including vendor and release.
  - The communications protocol (for example, TCP/IP or HLLAPI), including vendor and release.
  - The software release.
  - The server you are accessing, including release (for example, 5.2).
- The stored procedure (preferably with line numbers) or FOCUS commands being used in server access.
- The name of 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?
  - 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, WebFOCUS, 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 the code to access a single 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 Documentation Services staff welcomes any opinion you can offer regarding this manual. Please use the Reader Comments form at the end of this manual to relay suggestions for improving the publication or to alert us to corrections. You can also use the Documentation Feedback form on our Web site, <http://www.informationbuilders.com>.

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 World Wide Web site (<http://www.informationbuilders.com>) or call (800) 969-INFO to speak to an Education Representative.

---

---

# Contents

<b>1. Working in the Maintain Development Environment</b>	<b>1-1</b>
Opening WebFOCUS Maintain	1-2
Applications and Projects	1-2
What Are the Building Blocks of a WebFOCUS Maintain Project?	1-3
Creating New Project Files	1-4
Adding Existing Files to Your Project	1-6
What Other Types of Files Can Go Into a WebFOCUS Maintain Project?	1-6
Defining a Project's "Flow"	1-7
Editing Projects in the Maintain Development Environment	1-7
Viewing System Messages in the Output Window	1-10
Searching for Text in Your project	1-11
Incorporating Data Sources Into Your Project	1-12
Using WebFOCUS Procedures in Your Project	1-14
Using the Resource Wizard	1-16
<b>2. Developing Procedures</b>	<b>2-1</b>
Specifying Data Sources for Your Procedure	2-4
Using Functions in Procedures	2-6
Using Variables in Procedures	2-13
Using Data Source Stacks in Procedures	2-17
Using the Type Wizard	2-23
Passing Parameters Between Procedures	2-26
Using Variable Binding From WebFOCUS Reports	2-35
Using Import Modules	2-38
<b>3. Using the Form Editor</b>	<b>3-1</b>
Layout of the Form Editor	3-3
Using the Controls Palette	3-4
Using the Property Sheet	3-5
Using Drawing Aids	3-8
<b>4. Developing and Using Forms</b>	<b>4-1</b>
Before You Begin Designing Forms	4-2
Creating and Managing Forms	4-2
Exporting and Importing Forms	4-4
Changing Form Properties	4-4

- Dynamically Manipulating Forms at Run Time .....4-5
  - Displaying Forms at Run Time .....4-5
  - The Active Form .....4-7
  - The Non-Persistent Form .....4-7
  - Closing and Hiding Forms at Run Time .....4-7
  - Manipulating Form Properties .....4-9
  - Exiting Your Application at Run Time ..... 4-11
- Using a Driver Procedure ..... 4-11
- 5. Defining Events and Event Handlers .....5-1**
  - Using the Event Handler Editor .....5-2
  - Events .....5-5
    - Common Combinations of Events and Controls .....5-7
  - Using a Maintain Function as an Event Handler .....5-7
  - Using Script Functions as Event Handlers .....5-8
    - About JavaScript and VBScript .....5-9
    - Using Script Functions in Your Project .....5-9
    - Using Script Functions For Validation ..... 5-10
    - Managing Script Libraries ..... 5-17
    - Running and Debugging JavaScript and VBScript Functions ..... 5-18
  - Using Web Links as Event Handlers ..... 5-18
- 6. Developing and Using Controls .....6-1**
  - Which Control Should You Use? .....6-2
  - Manipulating Controls .....6-3
    - Selecting Multiple Controls .....6-3
    - Cutting, Copying, Pasting, and Duplicating Controls .....6-4
    - Resizing Controls .....6-6
    - Aligning Controls .....6-7
    - Spacing Controls ..... 6-10
    - Grouping Controls ..... 6-10
    - Changing the Order of Controls ..... 6-11
    - Undoing and Redoing Actions ..... 6-12
    - Layering Controls ..... 6-12
  - Using Buttons ..... 6-15
    - Changing Button Properties ..... 6-17
  - Using Check Boxes ..... 6-18
    - Changing Check Box Properties ..... 6-22
  - Using Combo Boxes and List Boxes ..... 6-23
    - Changing Combo Box or List Box Properties ..... 6-30
  - Using Edit Boxes and Multi-Edit Boxes ..... 6-31
    - Changing Edit or Multi-Edit Box Properties ..... 6-34
    - Using Prompted Edit Boxes ..... 6-35

Using Frames .....	6-38
Changing Frame Properties .....	6-39
Using Grids .....	6-40
Changing the Properties of a Read/Write Grid .....	6-46
Using Functions in the Read/Write Grid .....	6-47
Using Group Boxes .....	6-53
Changing Group Box Properties .....	6-54
Using HTML Objects .....	6-55
Changing HTML Object Properties .....	6-58
Using HTML Tables .....	6-59
Creating Links in HTML Tables .....	6-68
Changing HTML Table Properties .....	6-70
Using Images .....	6-71
Tips for Using Images in Your Applications .....	6-76
Changing Image Properties .....	6-76
Using Image Maps .....	6-77
Changing Images at Run Time Using ImageDown and ImageOver .....	6-80
Using Java Applets .....	6-81
Changing Java Applet Properties .....	6-85
Using Lines .....	6-86
Changing Line Properties .....	6-87
Menus .....	6-87
Changing Menu Bar Properties .....	6-92
Using Radio Buttons .....	6-93
Changing Radio Button Properties .....	6-96
Determining the Layout of Your Radio Buttons .....	6-97
Using Text .....	6-98
Changing Text Properties .....	6-99
Using Text Controls as Hyperlinks .....	6-100
Using ActiveX Controls .....	6-100
Changing ActiveX Control Properties .....	6-105
Unstable ActiveX Controls .....	6-106
Dynamically Manipulating Controls at Run Time .....	6-106
Defining Global Attributes .....	6-107
Defining Colors for Your Form and Controls .....	6-108
Assigning Help to Your Forms and Controls .....	6-110
Assigning Tab Order to Controls .....	6-111

<b>7. Developing Classes and Objects</b>	<b>7-1</b>
What Are Classes and Objects?	7-2
Class Properties: Member Variables and Member Functions	7-2
Inheritance: Superclasses and Subclasses	7-4
Defining Classes	7-4
Reusing Classes: Class Libraries	7-14
Declaring Objects	7-15
<b>8. Debugging WebFOCUS Maintain Applications</b>	<b>8-1</b>
Type on EDAPRINT	8-2
Maintain Statement Trace	8-2
Compiling Maintain Procedures	8-3
Enabling Server Tracing and Setting Trace Levels	8-5
Viewing Trace Output	8-6
Sample Usage Scenarios for Tracing	8-7
<b>9. Developing an Application for a Shared Application Server</b>	<b>9-1</b>
Why Use a Shared Application Server?	9-2
Writing Applications for the Shared Application Server	9-2
Designing Transactions for the Shared Application Server	9-3
Deploying and Testing Applications on Shared Application Servers	9-4
Preparing a Shared Application Server	9-4
<b>10. Managing Team Development</b>	<b>10-1</b>
Team Development	10-2
<b>11. Ensuring Transaction Integrity</b>	<b>11-1</b>
Why Is Transaction Integrity Important?	11-3
Defining a Transaction	11-3
When Does a Data Source Command Cause a Transaction to Fail?	11-4
Canceling a Transaction	11-5
Transactions and Data Source Position	11-5
How Large Should a Transaction Be?	11-5
Designing Transactions That Span Procedures	11-6
Designing Transactions That Span Data Source Types	11-7
Designing Transactions in Multi-Server Applications	11-8
When an Application Ends With an Open Transaction	11-8
Evaluating If a Transaction Was Successful	11-8
Concurrent Transaction Processing	11-9

Ensuring Transaction Integrity for FOCUS Data Sources .....	11-10
Setting COMMIT .....	11-11
Sharing Access to FOCUS Data Sources .....	11-12
How the FOCUS Database Server and Change-verification Work .....	11-13
Selecting Which Segments Will Be Verified for Changes .....	11-14
Identifying the FOCUS Database Server .....	11-15
Using Report Procedures and a FOCUS Database Server .....	11-16
Accessing Report Procedures When Using a FOCUS Database Server .....	11-16
Sharing Data Sources With Legacy MODIFY Applications .....	11-18
Ensuring Transaction Integrity for DB2 Data Sources .....	11-18
Using Transaction Locking to Manage DB2 Row Locks .....	11-20
Using Change Verification to Manage DB2 Row Locks .....	11-22
<b>12. Editing Project Components as Text .....</b>	<b>12-1</b>
Opening a Project Component .....	12-3
Typing Text .....	12-3
Performing Basic Editing Functions .....	12-3
Finding and Replacing Text .....	12-6
Sizing the Editor Window .....	12-7
Changing Colors, Fonts, and Tabs .....	12-7
Changing Text Color .....	12-9
Changing the Font Type, Style, and Size .....	12-10
Setting Tabs .....	12-10
Using Bookmarks to Move Within the Editor .....	12-10
Previewing and Printing Text .....	12-11
Using the Language Wizard .....	12-12
<b>13. Customizing the Maintain Development Environment .....</b>	<b>13-1</b>
Managing Windows .....	13-2
Using Toolbars .....	13-4
Customizing Toolbars .....	13-6
Using the Status Bar .....	13-11
Setting Preferences .....	13-12
Viewing the Tip of the Day .....	13-15

## *Contents*

---

---

## CHAPTER 1

# Working in the Maintain Development Environment

### Topics:

- Opening WebFOCUS Maintain
- Applications and Projects
- What Are the Building Blocks of a WebFOCUS Maintain Project?
- Editing Projects in the Maintain Development Environment
- Incorporating Data Sources Into Your Project
- Using WebFOCUS Procedures in Your Project
- Using the Resource Wizard

The Maintain Development Environment enables you to design, test, and deploy WebFOCUS Maintain projects.

This chapter describes how to open the Maintain Development Environment, how to create a WebFOCUS Maintain project (the umbrella under which an application's files are stored), the basic building blocks of a project, and a little bit about the various parts of the Maintain Development Environment.

## Opening WebFOCUS Maintain

---

WebFOCUS Maintain is an optional part of WebFOCUS Developer Studio, and you open it from the main Developer Studio window.

### **Procedure** How 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, can access varied data sources, and can 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 the \Ibi\Apps 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?**

---

Once you open the Maintain Development Environment and create a project, you will need to add components to your project. What kinds of components can you add to your project?

Basically, you can add any file you want to your project; although, we should point out that when you deploy your project to create an application, these files should make sense in the context of where you're deploying them to. For example, WebFOCUS Maintain can deploy to MVS machines; deploying a graphic file there might not work too well.

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

- **HTML Forms** are files with the extensions .htm and .html.
- **Maintain Files** are Maintain procedures and forms. You create them using the Maintain Development Environment. For more information on procedures, see Chapter 2, *Developing Procedures*.
- **Master Files** are files that describe the data that WebFOCUS Maintain will operate on and have the extensions .mas and .acx. For more information, see *Incorporating Data Sources Into Your Project* on page 1-12.
- **Procedures** are new FOCUS procedures that you create using WebFOCUS Developer Studio. For more information, see *Using WebFOCUS Procedures in Your Project* on page 1-14.

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

## Creating New Project Files

WebFOCUS Maintain enables you to create new project files for the following items:

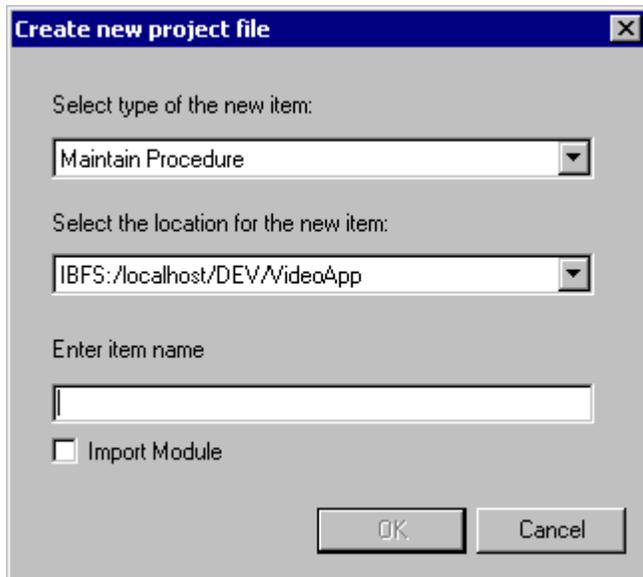
- Maintain procedure
- FOCUS procedure
- Master file
- VB Script
- Java Script
- HTML Document
- Text

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

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...*
  - 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.
3. Choose a location for the component. Your options include any of the directories in the project path.
4. Type a name for the component.
5. Click *OK*.

## Reference **New Project File Dialog Box**

You use the New project file dialog box to create new files for your project.



This dialog box contains the following fields:

### **Select type of the new item**

Contains a list of the most common items you are likely to add to your project.

### **Select the location for the new item**

Contains a list of all of the directories in your project's path. For more on paths, see *How to Change a Project's Path* on page 1-9.

### **Enter item name**

Enter the name for your new project component here. This name must conform to Windows naming standards. WebFOCUS Maintain will append the appropriate extension to the file name.

### **Import module**

(Maintain Procedures only.) Indicates that the Maintain procedure that you are creating should 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 Chapter 2, *Developing Procedures*.

## Adding Existing Files to Your Project

In addition to creating files from scratch, you can also easily add existing files to your project. WebFOCUS Maintain enables you to add many more types of files than you can create from scratch.

### **Procedure** How to Add an Existing File to Your Project

1. Make sure 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. Otherwise, see *How to View New File Types in the Explorer* on page 1-9.
2. Make sure that 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 to the project.
  - or
  - Using Windows Explorer, move the file to one of the directories that is in the project path.
3. Turn on Display all files in the project paths:
  - Click the *Display all files in the project paths*  button.
  - or
  - 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.

## 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 means that you 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. For more information, see Chapter 2, *Command Reference* in *WebFOCUS Maintain Language Reference*.
- 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 a Project's "Flow"

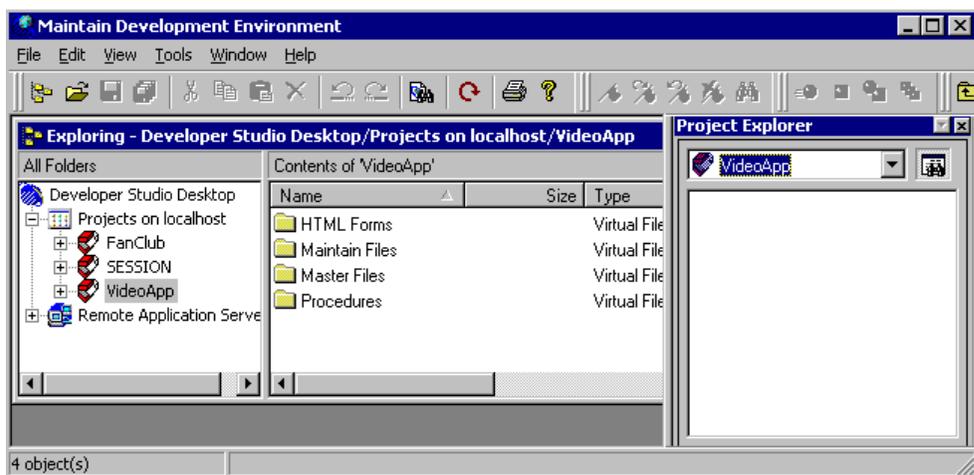
A project must contain one or more procedures. One of these procedures is your *starting procedure*, which is executed first when you execute the project. The other procedures are executed only if they are called explicitly using the CALL syntax.

A procedure is made up of functions (defined by the CASE and ENDCASE keywords). One of the functions in your procedure is called Top, and this function 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 need to 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*.

## Editing Projects in the Maintain Development Environment

When you open a new WebFOCUS Maintain project, your Maintain Development Environment will look something like the following:



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

### Explorer

(Usually on the left.) This window is where you manage your projects, their components, and the servers you can access.

For more information on using this window, see the *WebFOCUS Developer Studio Getting Started* manual.

### Project Explorer

(Usually on the right.) Contains another view of the Maintain procedures in your project. This window also has a *Display all files in the project paths* button.

If the project and any other component in the project 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 will see an asterisk next to the name of your project and the name of the component. To save a component in your project, do the following:

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 File toolbar.

## **Procedure** How to Save Your Project Automatically

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 a Project's Path

By default, a project's path 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:
  - a. 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*  and *Move item down*  buttons.
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 *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 your WebFOCUS Developer Studio documentation.

### **Procedure** How to View or Hide All Files in a Project's Path

- 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**

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**

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**

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

**Note:** The project component will exist, 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.

**Run**

Displays messages that occur while running a project. This includes error messages and TYPE statement output (the messages written to STDOUT in older versions of Cactus).

**Find**

Displays search results from the Search project command and enables you to open those project components that satisfy your search conditions.



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

In the View menu, click *Output window*.

### **Searching for Text in Your project**

You can easily search for text throughout your entire project using the Search project command. (You can also search for text in an individual project component such as a procedure while editing as text. For more information, see *Finding and Replacing Text* in Chapter 12, *Editing Project Components as Text*.)

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

1. In the Edit menu, click *Search project...*

or

Click the *Search project*  button on the General toolbar.

2. In the Search project dialog box, enter the text for which you want to search in the *Find What* box or use the text from a previous search by selecting it from the list.
3. If you want to match the case of the text for which you are searching, select *Match case*.
4. If you want to build more complex searches using regular expression syntax, select *Regular expression*.
5. Click *Find Now*.
6. If the Output window is not open, open it by clicking *Output window* in the View menu.
7. 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.

## Reference Search Project Dialog Box

The Search project dialog box enables you to search for text throughout a project.



### Find What

Enter the text for which you want to search, or use the text from a previous search by selecting it from the list.

### Match case

Select this option to match the case of the Find What text.

### Regular expression

Enables you to build more complex searches using regular expression syntax.

## Incorporating Data Sources Into Your Project

---

One of 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 either FOCUS or text format, an Access File is not necessary.

WebFOCUS refers to data source descriptions as Master Files.

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

### **Procedure** How to Edit a Data Source Description

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

WebFOCUS Maintain opens the data source in the Master File Editor. For more information, see the *Describing Data With WebFOCUS Language* manual.

**Note:** Editing the data source description does not change the actual data in the data source.

If you wish to edit the data source description as text, you can do so using Developer Studio.

### **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 will not show up in the Project Explorer unless you use it in a procedure. For more information, see *Specifying Data Sources for Your Procedure* in Chapter 2, *Developing Procedures*.

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

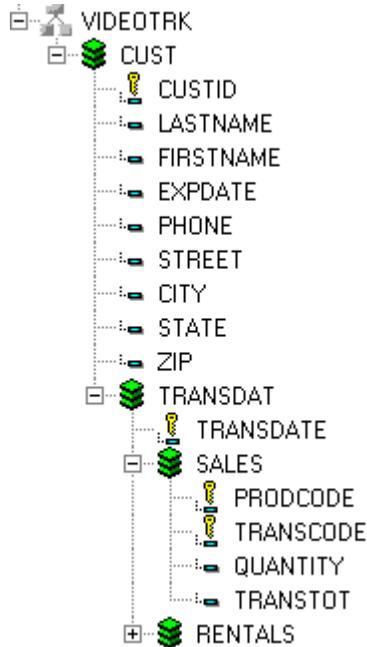
WebFOCUS Maintain will display the names of the segments in the data source.

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

WebFOCUS Maintain will display the names of the fields in the data source. Any key fields will 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 WebFOCUS Procedures in Your Project

---

One of the components your project can have is an *FOCUS procedure*. You commonly include FOCUS code (including Dialogue Manager commands) into your project to execute and process reports. (You can get the results of a FOCUS report in a Maintain data source stack, and then do further processing.)

If you have created any procedures using Developer Studio, you will see them in the External Procedures folder.

If your external procedure is a WebFOCUS report with static parameters, you can do one of the following:

- You can execute the report using EXEC and return the results to WebFOCUS Maintain using ON TABLE PCHOLD (only if the default format on the WebFOCUS Server is HTML) or ON TABLE HOLD FORMAT HTMTABLE. For more information, see *How to Display the Output From a FOCUS Report on a Form* on page 1-15 and *How to Use the Output From a FOCUS Report on a Form* on page 1-16.
- You can execute the report using a Web link. For more information, see *How to Use the URL Wizard to Define an HTTP Web Link* in Chapter 5, *Defining Events and Event Handlers*.

If you are using a FOCUS Database Server and will be accessing Maintain procedures and external report procedures on the same WebFOCUS Server, you need to access them in a particular way to ensure the integrity of your logical transactions. For more information, see Chapter 12, *Editing Project Components as Text*.

### **Procedure** How to Edit the Source Code of a FOCUS Procedure

1. Right-click the procedure.
2. In the shortcut menu, click *Open*.

WebFOCUS Maintain opens the Text Editor where you can make changes to your procedure.

However, we recommend that you return to WebFOCUS Developer Studio, which contains a full suite of tools for editing your FOCUS procedure.

### **Procedure** How to Display the Output From a FOCUS Report on a Form

1. Make sure the report contains the following line:  
`ON TABLE PCHOLD FORMAT HTMTABLE`
2. Include the report as an external 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 INTO stack;`
5. Display the contents of the stack using an HTML Object.

For more information on FOCUS reports, see your FOCUS documentation.

### **Procedure** How to Use the Output From a FOCUS Report on a Form

1. Make sure the report contains the following line:

```
ON TABLE PCHOLD
```

2. Include the report as an external procedure in your project.
3. Create a stack with columns that match the names of the columns in the report.

**Tip:** 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 report procedure using the following syntax:

```
EXEC procedure INTO stack;
```

5. Display the contents of the stack using anything but an HTML Object.

**Note:** The default format of the WebFOCUS Server on which the report is executed must be HTML.

For more information on FOCUS reports, see your FOCUS documentation.

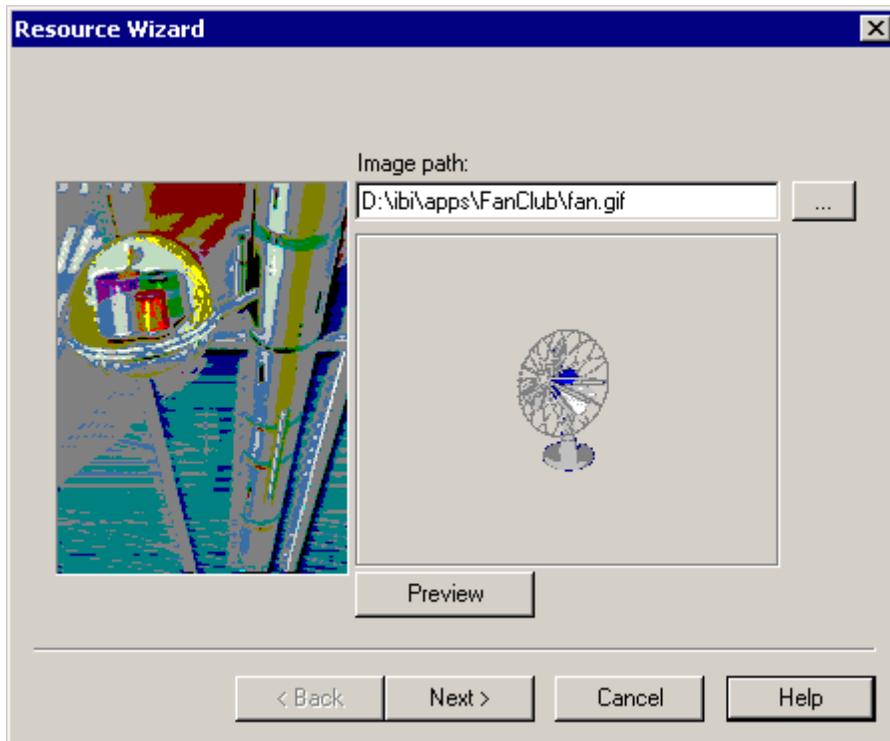
## **Using the Resource Wizard**

---

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.

**Reference Resource Wizard: Specify Path for Image**

You use this dialog box to specify a path to an image.



This dialog box contains the following options:

**Image path**

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



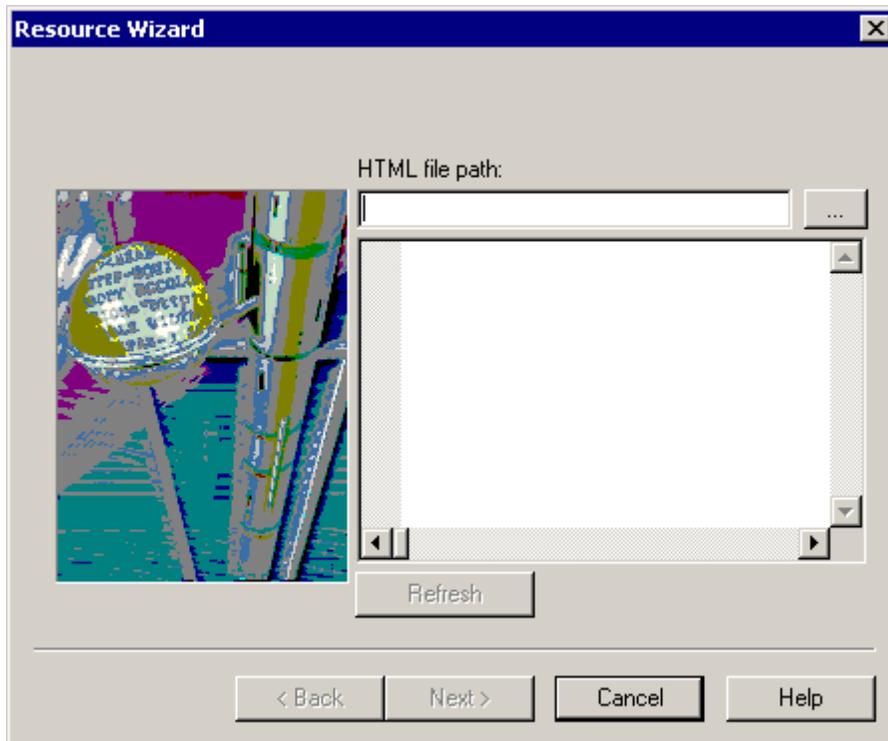
Opens the Open dialog box, where you can search for your image file.

**Preview**

Enables you to view an image whose path you just typed.

## **Reference** Resource Wizard: Specify Path for HTML File

You use this dialog box to specify a path to an HTML file.



This dialog box contains 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 just type a path and file name here.



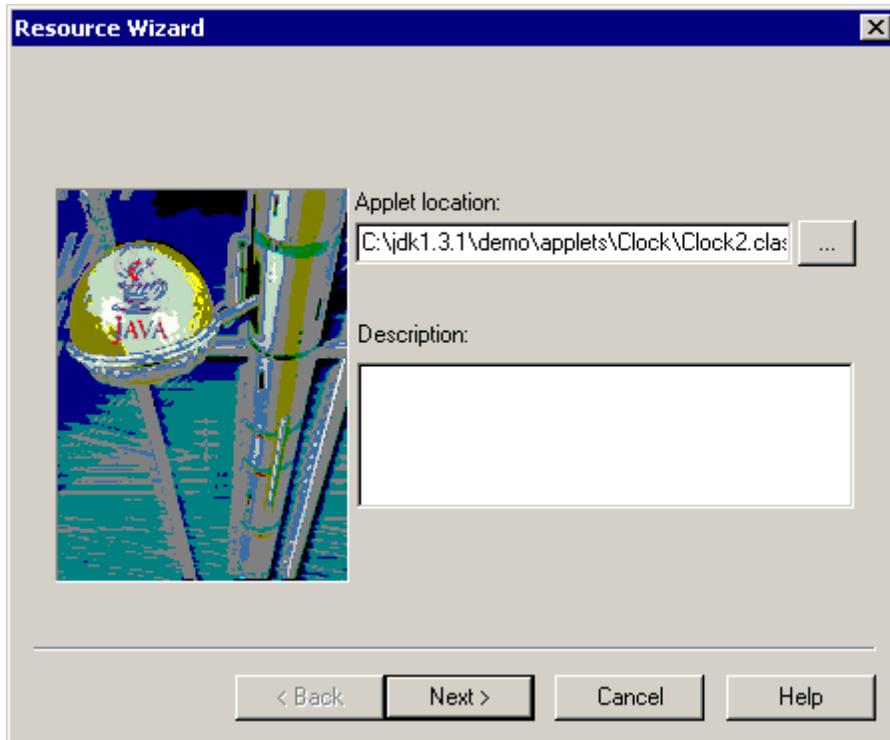
Opens the Open dialog box, where you can search for your HTML file.

### **Refresh**

Enables you to view an HTML file whose path you just typed.

**Reference Resource Wizard: Specify Path for Java Applet**

You use this dialog box to specify a path to a Java applet.



This dialog box contains the following options:

**Applet location**

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



Opens the Open dialog box, where you can search for your Java applet.

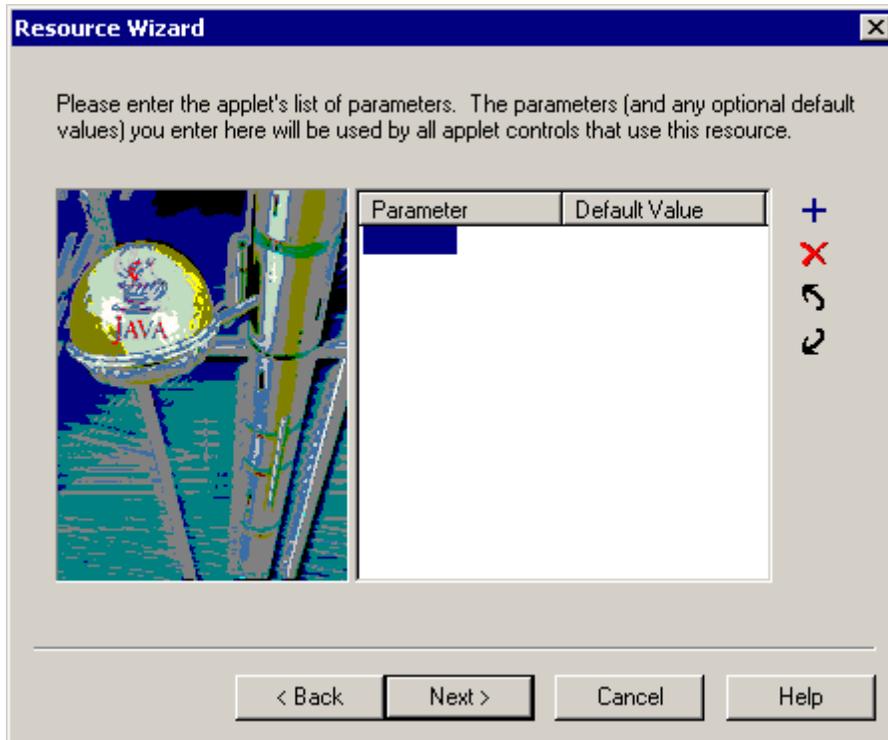
**Description**

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.

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



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



Deletes the selected parameter.



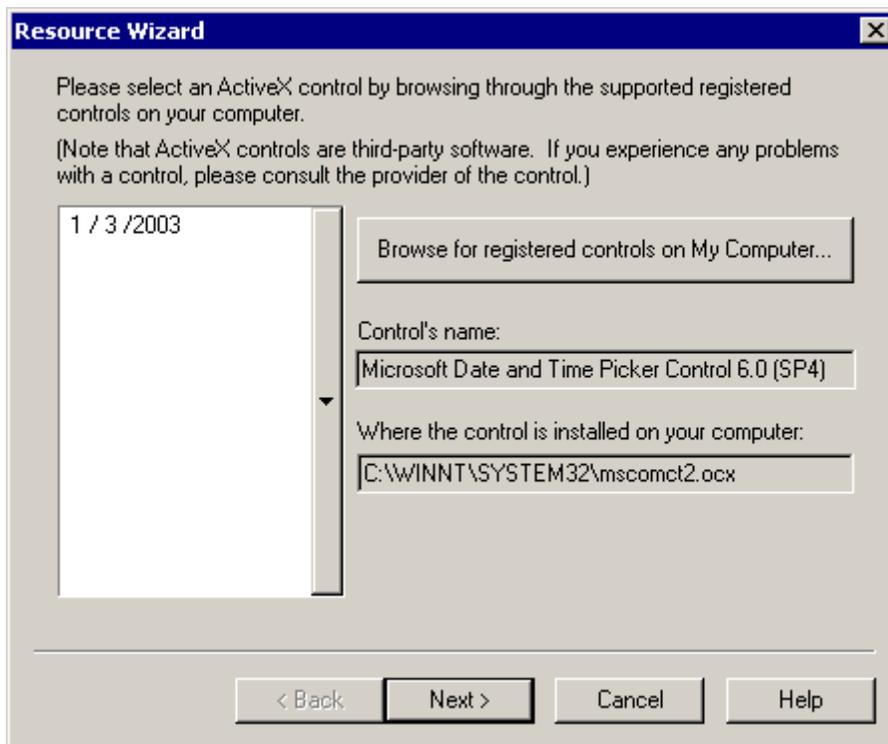
Moves a selected parameter up in the list of parameters.



Moves a selected parameter down in the list of parameters.

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

The Resource Wizard 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 is displayed on the screen by default. If the ActiveX control you want does not appear on the list, you need to download it to your computer.



#### **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**

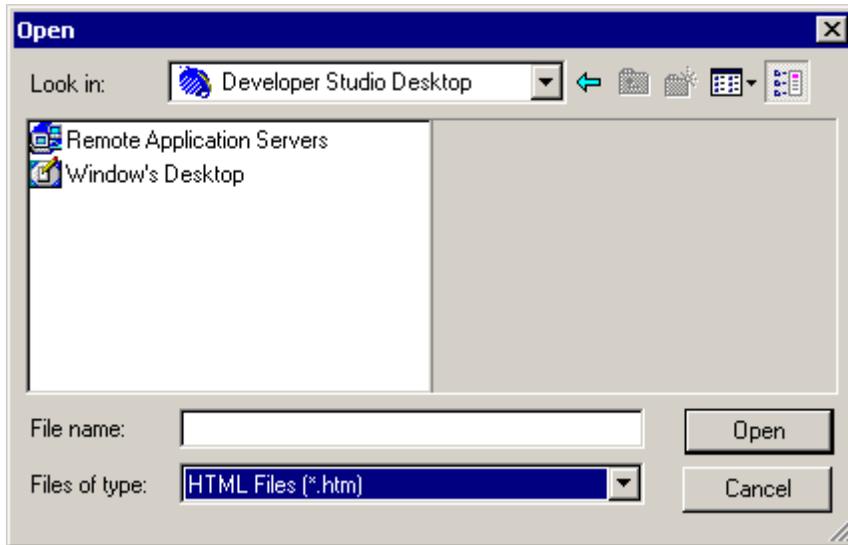
Displays the name of the ActiveX control selected from the Insert ActiveX Control dialog box.

### Where the control is installed on your computer

Displays where the ActiveX control resides on your computer.

### Reference **Open Dialog Box**

You use the Open dialog box 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.



This dialog box contains the following:

#### **Look in**

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

#### **Remote Application Servers**

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

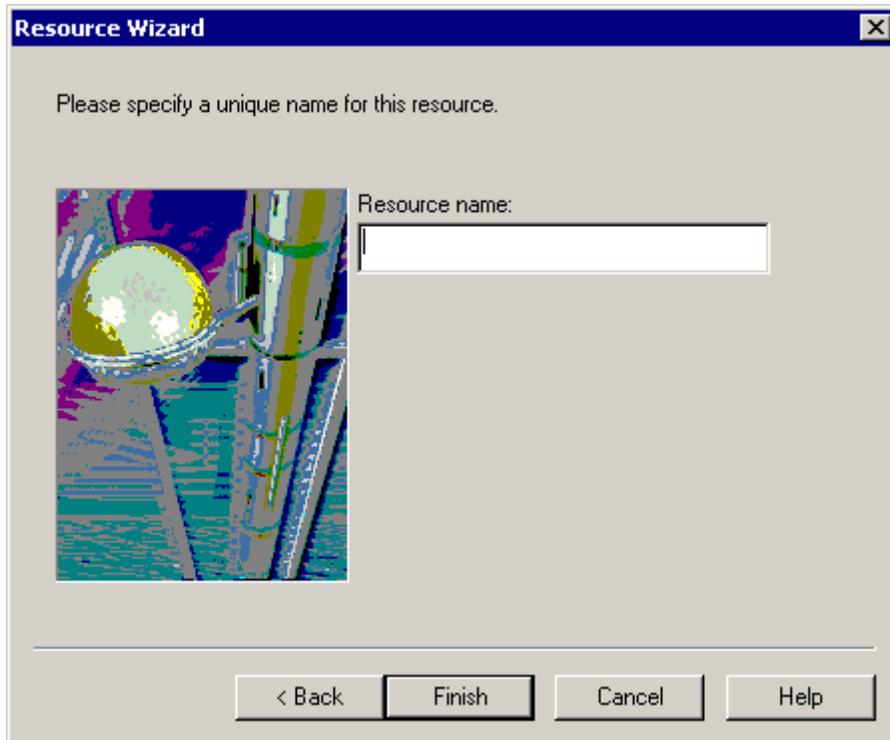
#### **Windows Desktop**

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

**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. 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

You will see this dialog box 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 will ensure that the resource gets deployed to the proper location. Select *Don't show this again* if you don't need WebFOCUS Maintain to display this dialog box.



This dialog box will also warn you if you are overwriting a resource with the same name. You can either rename the resource or overwrite the existing one.



---

---

## CHAPTER 2

# Developing Procedures

### Topics:

- Specifying Data Sources for Your Procedure
- Using Functions in Procedures
- Using Variables in Procedures
- Using Data Source Stacks in Procedures
- Using the Type Wizard
- Passing Parameters Between Procedures
- Using Variable Binding From WebFOCUS Reports
- Using Import Modules

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

This topic provides information on creating, editing, and viewing procedures. There is also 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, using the Type Wizard to assign data types, and passing parameters between procedures.

When you develop your project, you will divide it into a few logical components that accomplish specific tasks, such as placing a new order and 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* in Chapter 1, *Working in the Maintain Development Environment*. For information on how to add an existing procedure to your project, see *How to Add an Existing File to Your Project* in Chapter 1, *Working in the Maintain Development Environment*.

Procedures can operate on data sources. If a procedure accesses a data source, either to extract data or to write data, then you must specify which data sources the procedure accesses. For more information, see *Specifying Data Sources for Your Procedure* on page 2-4.

A procedure can be made up of any combination of the following:

- Functions
- Variables
- Forms (see Chapter 4, *Developing and Using Forms*)
- Data source stacks
- Classes (see Chapter 7, *Developing Classes and Objects*)

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 *Passing Parameters Between Procedures* on page 2-26.

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 be your starting procedure.
3. Click *OK*.

## **Procedure** How to Edit a Procedure's Source Code

WebFOCUS Maintain generates Maintain language code for all of the 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 any changes you wish to the code between these two keywords. For more information, see *Command Reference* in *WebFOCUS Maintain Language Reference*.
4. Close the Procedure Editor.

**Tip:** You can add Maintain language code to your procedure 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 easily obtain help on any Maintain language keyword by selecting the keyword or placing the insertion point in the keyword and pressing F1.

**Note:** If WebFOCUS Maintain determines that you have a syntax error in your procedure, you will see the words "Syntax Error" next to the procedure name in the Project Explorer, an error message will appear in the Output window, and the offending line will be marked with an exclamation mark in the Procedure Editor. You will be unable to edit forms or perform drag-and-drop functions in that procedure until you fix the syntax error. You can easily open the procedure at the syntax error by double-clicking on the syntax in the Output window.

## **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.
2. Press F4.

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 have 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 from the project's Data Sources folder to access.

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, you specify the data sources to use in your procedure after the *MAINTAIN FILE* keywords. For more information, see *MAINTAIN* in Chapter 2, *Command Reference* in *WebFOCUS Maintain Language Reference*.

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 Chapter 1, *Working in the Maintain Development Environment*.

### **Procedure** How to Specify Data Sources to Use 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  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.

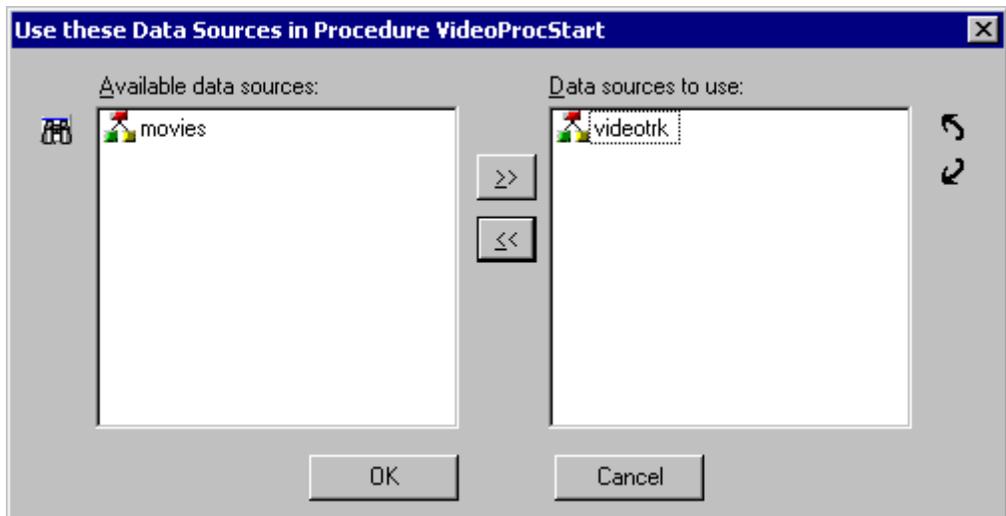
**Tip:** If you do not see the data source you want to use, then you need to make sure that it is in the project paths. For more information, see *How to Add an Existing File to Your Project* in Chapter 1, *Working in the Maintain Development Environment*.

4. To remove data sources you no longer need, select them in the *Data sources to use* list and click the  button.
5. Use the *Move down* and *Move up* buttons to change the order of the data sources. (The only time the data source order matters is when data sources have identical fieldnames 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 will add it to the project.

### Reference Use these Data Sources in Procedure Dialog Box

You specify the data sources to use in your procedure with the Use these Data Sources in Procedure dialog box.



This dialog box contains the following fields:

#### 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 will use.



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 logic allowed outside of a function is variable declarations, object declarations, and class definitions (CASE, DECLARE, DESCRIBE, END, MAINTAIN, and MODULE statements).

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

Some functions also have a *return value*—that is, they calculate a value and return this value to whatever called the function. Functions that are *event handlers* (that is, they 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 *Local and Global Declarations* in Chapter 2, *Command Reference* in *WebFOCUS Maintain Language Reference*.

You can also call another function from within a function.

The combination of a function's arguments and return value 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  button to open the Function Return dialog box where you can define it.
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.

**Caution:** 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 easily obtain help on any Maintain language keyword by selecting the keyword or placing your insertion point in the keyword and pressing F1.

10. Close the Procedure Editor.

### **Procedure** How to Edit a Function's Name, Arguments, Return Value, and Description

1. Right-click the function in the Project Explorer.
2. In the shortcut menu, click *Edit...*
3. Make any necessary 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 any changes you wish to the code between the *CASE functionname* and *ENDCASE* keywords. For more information, see *Command Reference* in Chapter 2, *Command Reference* in *WebFOCUS Maintain Language Reference*.
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 easily obtain help on any Maintain language keyword by selecting the keyword or placing your insertion point in the keyword and pressing F1.

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

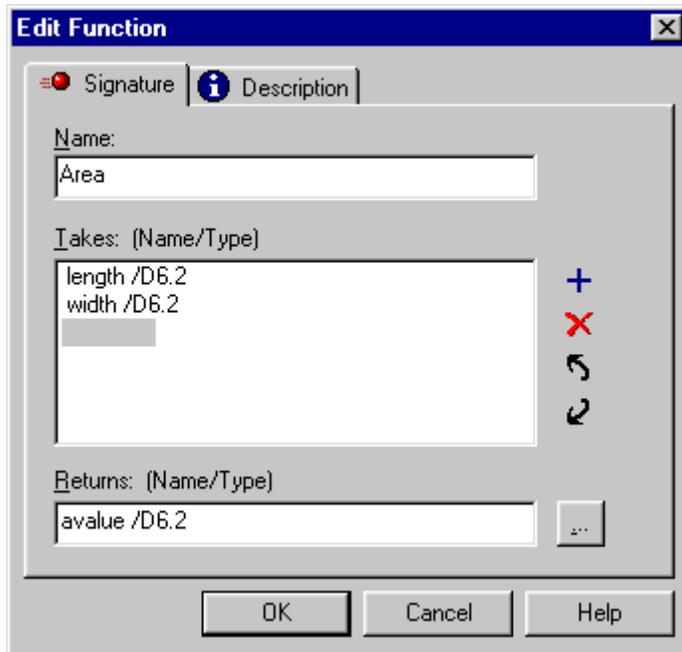
1. In the Project Explorer, make sure you can see both the function and the procedure you want to move it to.
2. Select the function and drag it to the procedure you want to move it to.

**Procedure** **How to Copy a Function to Another Procedure**

1. In the Project Explorer, make sure you can see both the function and the procedure you want to move it to.
2. Select the function, hold down the Ctrl key, and drag the function to the procedure you want to move it to.

## 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.



This dialog box has the following elements:

### **Name**

Contains 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 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. In fact, 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.

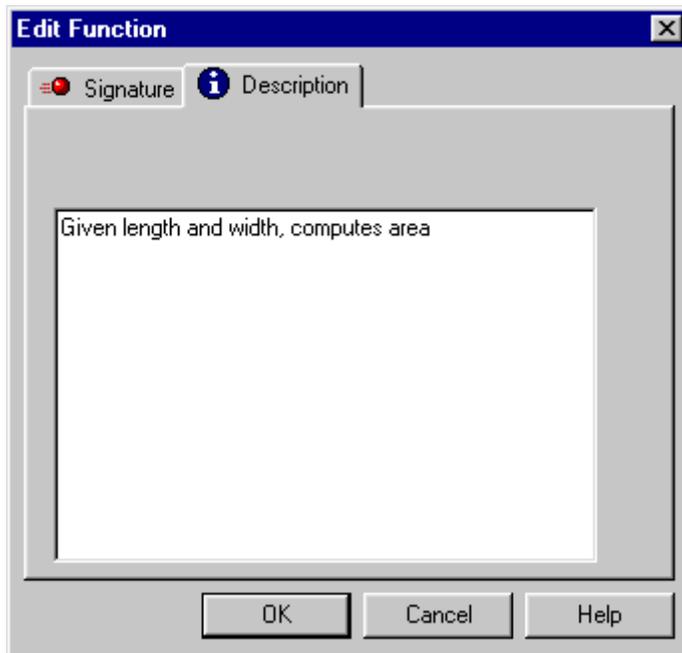


Opens the Function Return dialog box where you can define the return value if you do not want to type it in.

## 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.

You can easily 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 2-3.

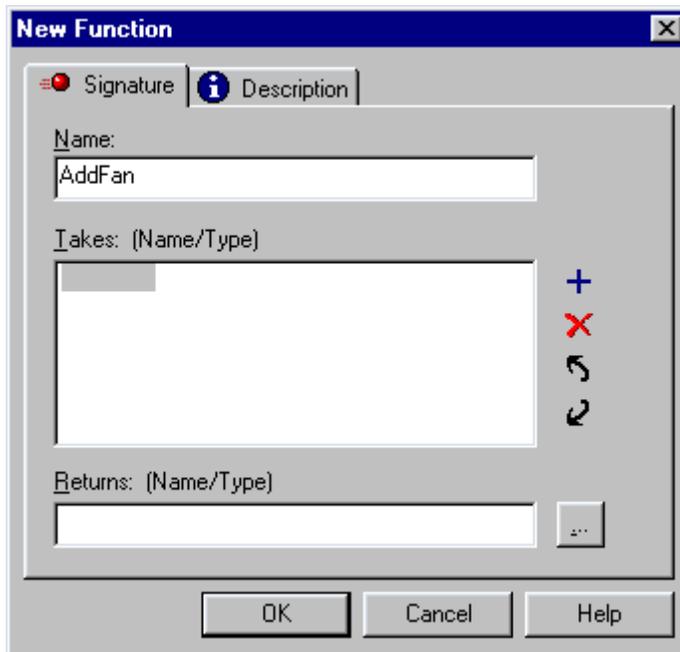


## Example **Creating a Function**

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*.



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*.

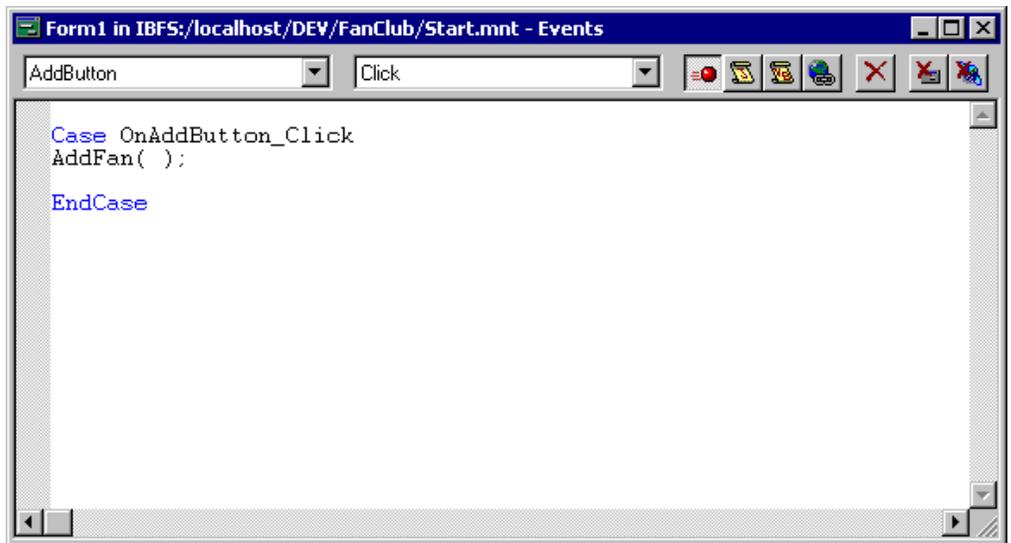
7. Open the form you will execute this function from.
8. Create a button named *AddButton* with the text *Add*.



9. Double-click the *Add* button to open the Events dialog box.

10. Select *Click* from the list of events.
11. Make sure 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 will look like the following:



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  button to open the Type Wizard.
  - 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*; that is, it is available to any code written in this procedure. To use it in another procedure, you will need to pass it to the other procedure. For more information, see *Passing Parameters Between Procedures* on page 2-26.

### **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 a Variable's Source Code

WebFOCUS Maintain generates Maintain language code for variables. If you wish, 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 any changes you wish to the code after the *DECLARE* *variablename* keyword and before the semicolon. For more information, see Chapter 2, *Command Reference* in *WebFOCUS Maintain Language Reference*.
4. Close the Procedure Editor.

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

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

1. In the Project Explorer, make sure you can see both the variable and the procedure you want to move it to.
2. Select the variable and drag it to the procedure you want to move it to.

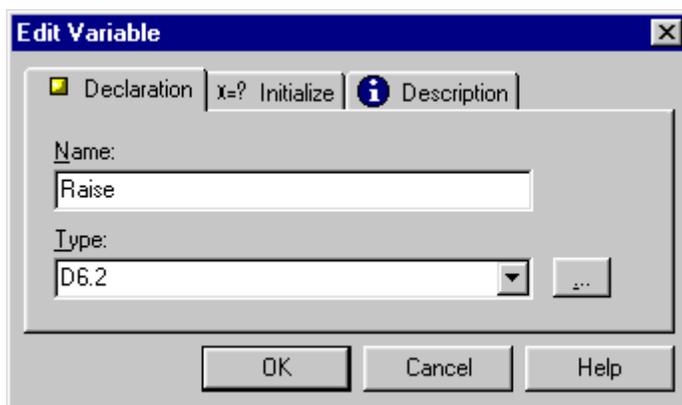
### **Procedure** How to Copy a Variable to Another Procedure

1. In the Project Explorer, make sure you can see both the variable and the procedure you want to move it to.
2. Select the variable, hold down the Ctrl key, and drag the variable to the procedure you want to move it to.

### **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, or 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, which looks exactly like the Edit Variable dialog box without the x=? Initialize tab.



This dialog box has the following components:

**Name**

Contains the name of the variable or 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 the specification for the data type.

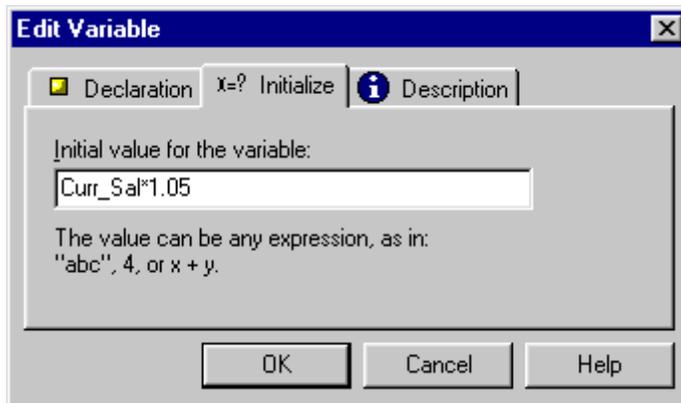


Opens the Type Wizard, where you can specify a new data type for the argument or return value. For more information, see *Using the Type Wizard* on page 2-23.

**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 *Packed-Decimal Format* and *Character Formats* in *Describing Data With WebFOCUS Language* manual.

**Reference Variable Editor: x=? Initialize Tab**

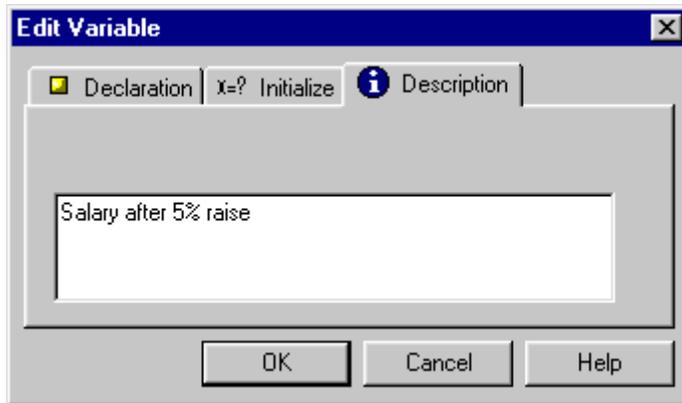
When you are creating or editing a variable, use the x=? Initialize tab in the New Variable or Edit Variable dialog box to enter a value for your variable. This value can be any expression; see *Expressions Reference* in *WebFOCUS Maintain Language Reference*.



## 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.

You can easily view the description of a variable in the Project Explorer with the Show description command. For more information, see *How to View the Description of a Procedure Component* on page 2-3.



## 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 to 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 editing the procedure as text and typing the appropriate syntax. A stack’s columns are usually based on actual data source fields, but you can also add your own columns.

Then you 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. (You can 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 2-18. For more information on the INFER command, see *Command Reference* in *WebFOCUS Maintain Language Reference*.
- **Implicitly**, by loading data into the data source stack using the NEXT or MATCH commands. *NEXT* in *Command Reference* *MATCH* in *Command Reference* For more information on loading data into the data source stack, see *Command Reference* in *WebFOCUS Maintain Language Reference*.

**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. For more information, see the information on data source navigation in *Data Source Navigation Using NEXT: Overview* in *Command Reference* in *WebFOCUS Maintain Language Reference*.

### **Procedure** How to Create a Data Source Stack Explicitly Using the Stack Editor

1. Make sure 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 2-4.)
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  button.

**Tip:** If no data sources appear in the Available fields list, you need to specify which ones to use in your procedure. For more information, see *How to Specify Data Sources to Use for Your Procedure* on page 2-4.

**Note:** When you copy one of the fields in a segment into the definition of a stack, WebFOCUS Maintain automatically copies all of the rest of the fields in that segment and the key fields from the ancestor segments. These columns are called *implied columns*.

- **Create custom (user-defined) columns.** (You 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. If you open it, you will 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 you want to create the data source stack in. (WebFOCUS Maintain will prompt 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 haven't specified it already.

## **Procedure** How to Edit a Data Source Stack's Source Code

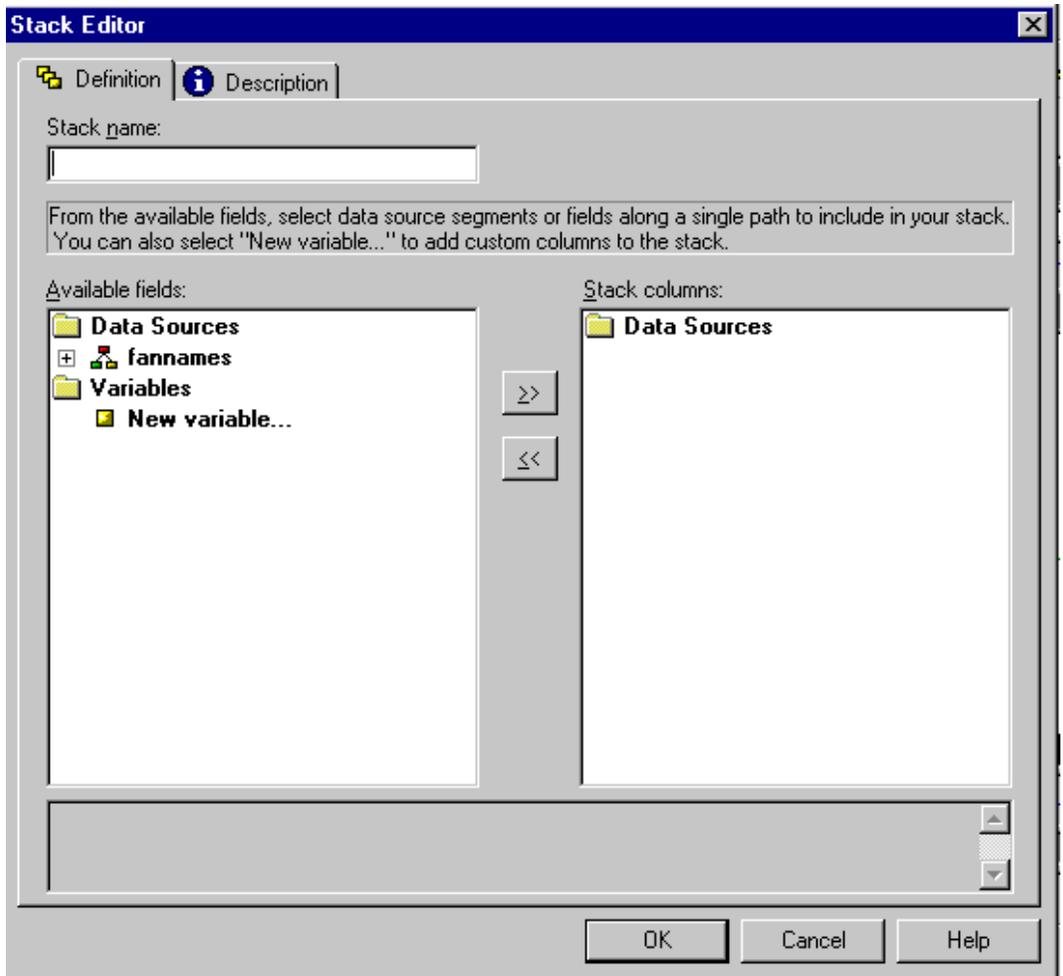
WebFOCUS Maintain generates Maintain language code for data source stacks. If you wish, you can edit this code directly in the Procedure Editor.

1. Right-click the data source stack in the Project Explorer.
2. In the shortcut menu, click *Edit Source*.
3. Make any changes you wish to the code in the INFER statement. For more information, see *INFER* in Chapter 2, *Command Reference* in *WebFOCUS Maintain Language Reference*.
4. Close the Procedure Editor.

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

## Reference **Stack Editor: Definition Tab**

When you create or edit a data source stack, WebFOCUS Maintain opens the Stack Editor dialog box.



This dialog box has the following fields:

### **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.

Notice that 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.

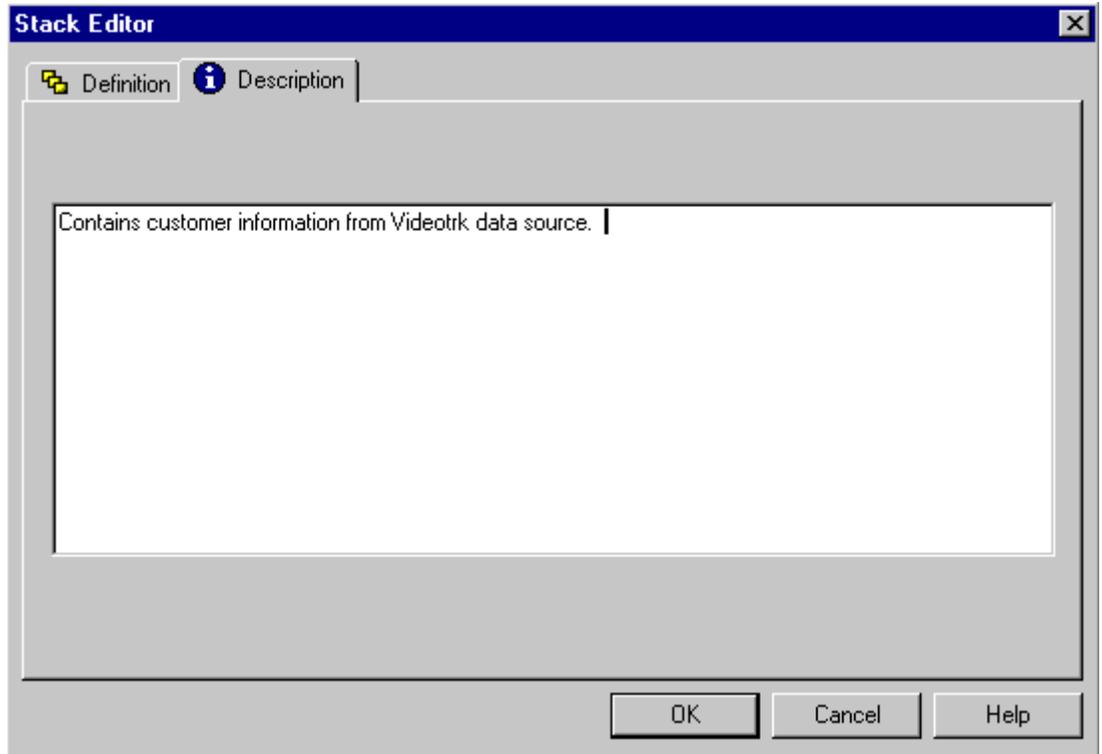
### **Infer datasource.segment.field into stack; and/or Compute stack.column/format;**

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

**Reference Stack Editor: Description Tab**

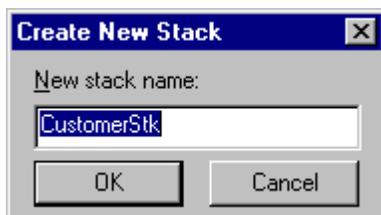
When you are creating or editing a data source stack, use the *Description* tab in the Stack Editor dialog box to enter a description.

You can easily view the description of a data source stack in the Project Explorer with the *Show description* command. For more information, see *How to View the Description of a Procedure Component* on page 2-3.



**Reference Create New Stack Dialog Box**

You use the Create New Stack dialog box to name a data source stack that you have created by dragging a segment from a data source to a procedure.



This dialog box has the following:

**New stack name**

Contains a proposed name for your new data source stack (the name of the segment, 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 need to assign a *data type* to it. A data type is either a *format* or a *class*. You 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.** Used 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.** Whole numbers or fractions, using 4 bytes of storage and a maximum number of 9 positions.
  - **Integer.** 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. See Chapter 7, *Developing Classes and Objects* for more information.

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



**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* (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, you will also need to 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:

<b>Suppresses zeros</b>	Suppresses leading zeros; displays a space if the value is 0.
<b>Leading zeros</b>	Adds leading zeros to the full field length.
<b>Include commas</b>	Inserts a comma after every third significant digit. This option takes effect automatically for Double Precision format.
<b>Floating dollar</b>	Adds a dollar sign immediately to the left of the number and inserts a comma after every third significant digit.
<b>Fixed dollar</b>	Adds a dollar sign to the left of the field and inserts a comma after every third significant digit.
<b>Scientific notation</b>	Displays the number in scientific notation.
<b>Credit negatives</b>	Adds the characters CR after a negative number.
<b>Bracket negatives</b>	Encloses a negative number in parentheses.

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 class will be available only if you have 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 Chapter 7, *Developing Classes and Objects*.

## **Passing Parameters Between Procedures**

---

All variables and data source stacks are defined only in the context of a procedure (or sometimes only in the context of a function in a procedure). However, when one procedure calls another, WebFOCUS Maintain enables you to pass variables and data source stack values back and forth between the two procedures:

- In the calling procedure, you specify the variables and data source stacks that you are passing to the called procedure and the variables and data source stacks that will be returned by the called procedure in the CALL statement.
- In the called procedure, you specify the variables and data source stacks whose values will be passed (as input parameters) and that will be passed back (as output parameters). You do this in the Procedure Parameters dialog box.

When you specify input and output parameters for a procedure, WebFOCUS Maintain adds a Parameters folder to the list of options under the procedure and lists the input and output parameters here.

**Note:** You cannot pass variables or data source stacks using the A0 format or Stack of format.

For more information on how to specify input and output parameters for the calling procedure, see the *CALL Command* in *Command Reference* in *WebFOCUS Maintain Language Reference*. For more information on calling procedures, see *Executing Other Maintain Procedures* in *WebFOCUS Maintain Concepts in Getting Started*.

**Procedure How to Specify Parameters for a Called Procedure**

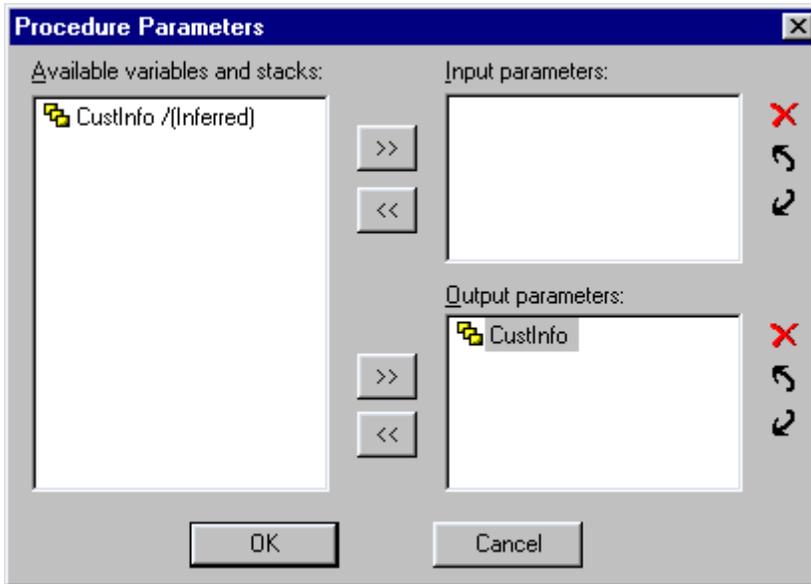
1. Define the variables and data source stacks whose values will be passed to the procedure and that the procedure will pass back. For more information, see *How to Create a Variable in a Procedure* on page 2-14 and *How to Create a Data Source Stack Explicitly Using the Stack Editor* on page 2-18.
2. 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  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  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, you will see your parameters listed in the MAINTAIN line of code. The input parameters will be after the FROM keyword; the output parameters will be after the INTO keyword. For more information on the MAINTAIN syntax, see Chapter 2, *Command Reference* in *WebFOCUS Maintain Language Reference*.

## Reference 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. You use this dialog box to define variables and data source stacks whose values this procedure is passing to a calling procedure and that this procedure is receiving from a calling procedure.



This dialog box has the following options:

### Available parameters and stacks

Lists the variables and data source stacks that you have defined in this procedure. These are the potential input and output parameters.



Copies a selected parameter or stack into the list of input or output parameters.



Removes a selected parameter or stack from the list of input or output parameters.

### Input parameters

Lists the parameters that you have copied here from Available parameters and stacks. When calling the procedure, you must supply values for these parameters.

**Output parameters**

Lists the parameters that you have copied here from Available parameters and stacks. Executing the procedure will set values for these parameters.



Deletes a selected parameter from the list of input or output parameters.



Moves a selected parameter up in the list of input or output parameters.



Moves a selected parameter down in the list of input or output parameters.

**Example Passing Parameters Between Two Procedures**

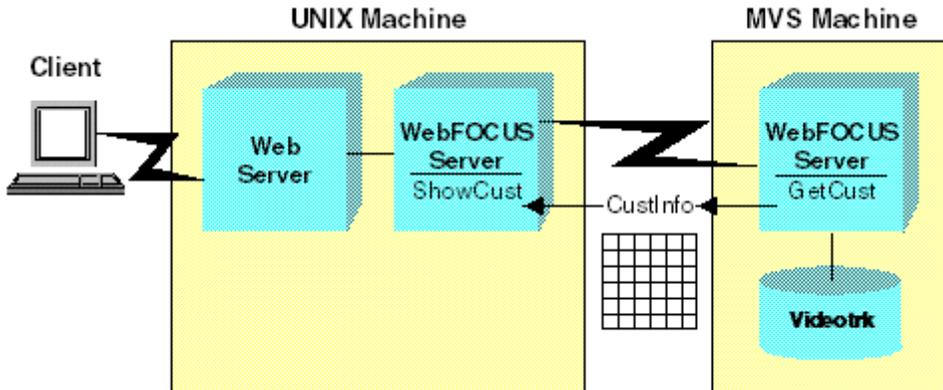
This example demonstrates how to pass a stack from a called procedure to a calling procedure, and the steps you must go through in order for both procedures to work properly.

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, and they need to call each other and pass data back and forth. (This is actually a very common occurrence in WebFOCUS Maintain.)

Let's suppose that the project (called VideoApp) needs to 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 Videotrck data source and places it in a data source stack called CustInfo.
3. GetCust passes CustInfo back to ShowCust.

4. ShowCust then displays a form with the information in CustInfo.



**Before you begin...**

Create a project named VideoApp, with the data source Videotrk (one of the sample data sources distributed with WebFOCUS Maintain).

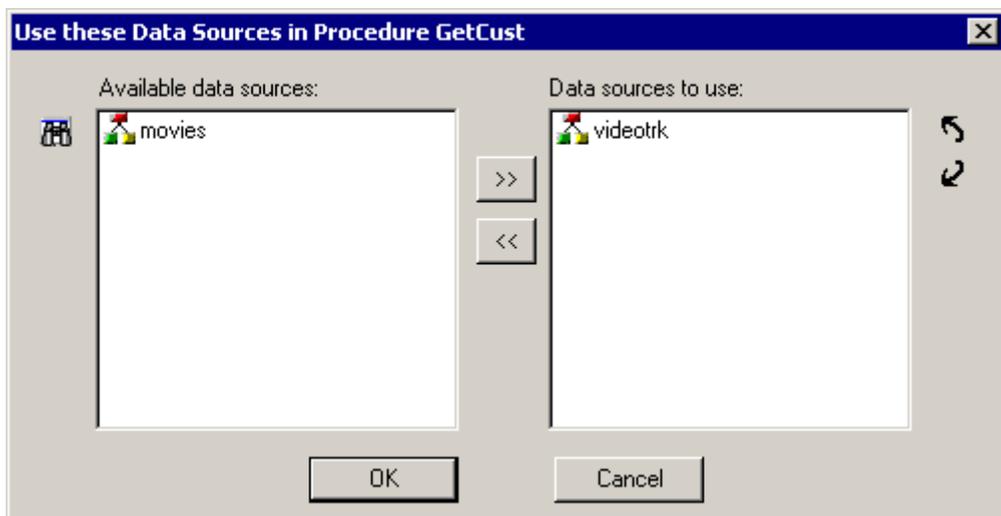
**Create GetCust:**

1. Right-click the *Maintain Files* folder in the Explorer window.
2. In the shortcut menu, click *New*, then click *Project file...*
3. Type *GetCust* and press Enter.

**Specify that GetCust 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, select *Videotrk* in the Available data sources list.

4. Click the  button.



5. Click *OK*.

**Create the data source stack *CustInfo* and load 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 could 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*.

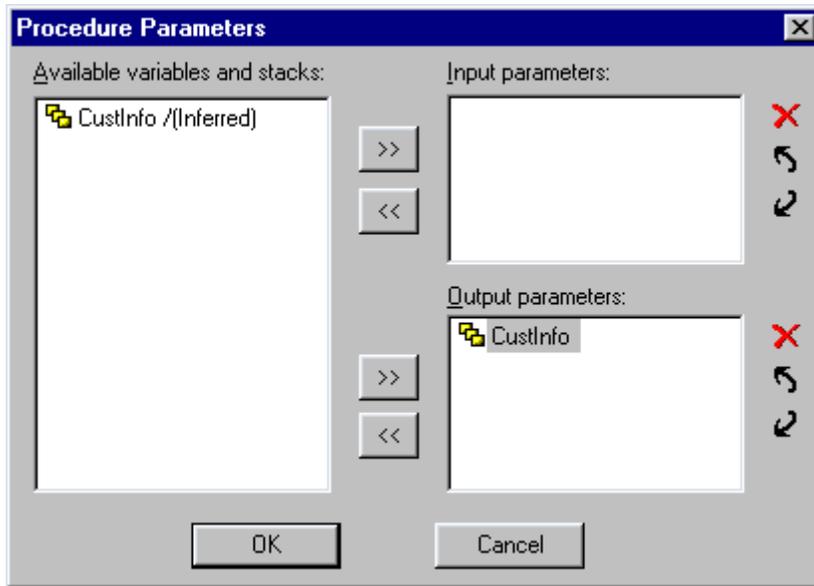
3. Close *GetCust*.

Notice that you created the stack *CustInfo* implicitly when you loaded data into the stack. For more information on stacks, see *Using Data Source Stacks in Procedures* on page 2-17.

**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, select *CustInfo* in the Available variables and stacks list.

4. Click the  button to copy CustInfo into the Output parameters list.



5. Click *OK*.
6. Notice that WebFOCUS Maintain creates a Parameters folder under GetCust and adds CustInfo there. The red arrow pointing down indicates that CustInfo is an output parameter.

**Create ShowCust:**

1. Right-click the *Maintain Files* folder in the Explorer window.
2. In the shortcut menu, click *New*, then click *Project file...*
3. Type *ShowCust* and press Enter.

**Specify that ShowCust should use the Videotrk data source:**

1. Right-click *ShowCust*.
2. In the shortcut menu, click *Use data sources...*
3. In the Use these Data Sources in Procedure ShowCust dialog box, select *Videotrk* in the *Available data sources* list.
4. Click the  button.
5. Click *OK*.

**Define CustInfo:**

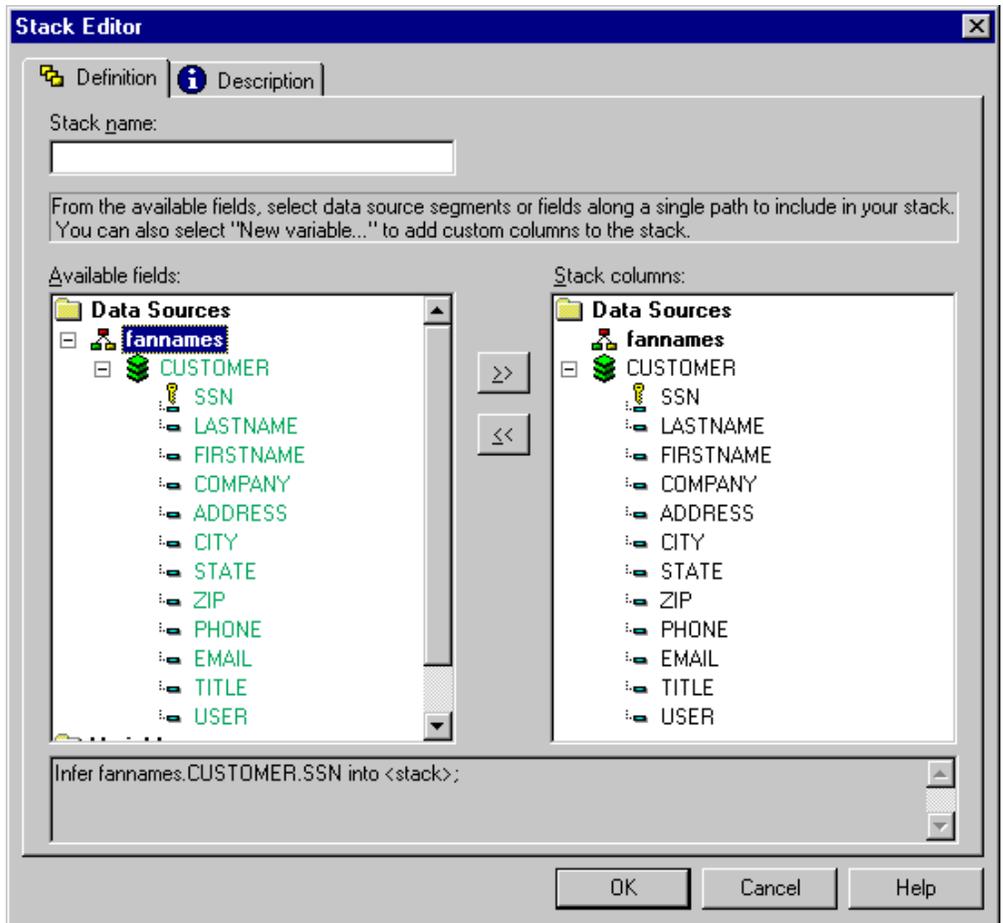
Since CustInfo is a data source stack in GetCust, ShowCust has no idea what the structure of CustInfo is. Therefore, you must define a stack here to receive its value. However, since you are not loading data into it here (you are getting the data from GetCust), all you need to do is define the columns.

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, we 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 Videotrk data source in the Available fields list.
6. Expand the CUST segment.
7. Select the *CUSTID* field.

8. Click the  button. (Notice that WebFOCUS Maintain copies all of the fields in the CUST segment into your stack definition along with CUSTID.)



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`

**Tip:** You could 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 Videotrck data source is now available to you in ShowCust to display to the end user.

Here is the Maintain code for ShowCust:

```
MAINTAIN FILE Videotrck
$$Declarations
Case Top
Infer Videotrck.CUST.CUSTID into CustInfo;
Call GetCust Into CustInfo ;
EndCase
END
```

Here is the Maintain code for GetCust:

```
MAINTAIN FILE Videotrck INTO CustInfo
$$Declarations
Case Top
For all next Videotrck.CUST.CUSTID into CustInfo ;
EndCase
END2
```

## Using Variable Binding From WebFOCUS Reports

---

You can easily link reports developed in WebFOCUS Developer Studio to Maintain procedures using the drill down to Maintain Procedure feature. One option provided by drill down is the ability to pass parameters from the WebFOCUS report to the Maintain procedure. For example, an end user could click on a row in a WebFOCUS report, which would then run 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 the 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, we call passed name/value pairs *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 have already created a WebFOCUS report that has a drill-down to the Maintain procedure. For more information on creating WebFOCUS reports, see *WebFOCUS Creating Reports With Graphical Tools*.

The general procedure is as follows:

1. Create a report with Report Painter.
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 that the report will be calling. This name must match the name of the Maintain procedure you will be creating!
6. Add the parameters that will be passed to the Maintain procedure.

Once you have created your report and specified 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 click *Import WebFOCUS Parameters*.
4. In the Open dialog box, select the WebFOCUS procedure you want to import parameters from 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 (for more information, see *Using the Type Wizard* on page 2-23) and click *OK*.

WebFOCUS Maintain places the following code at the beginning of your procedure:

```
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 make sure that the parameter name matches the variable name that you are using elsewhere in the procedure.

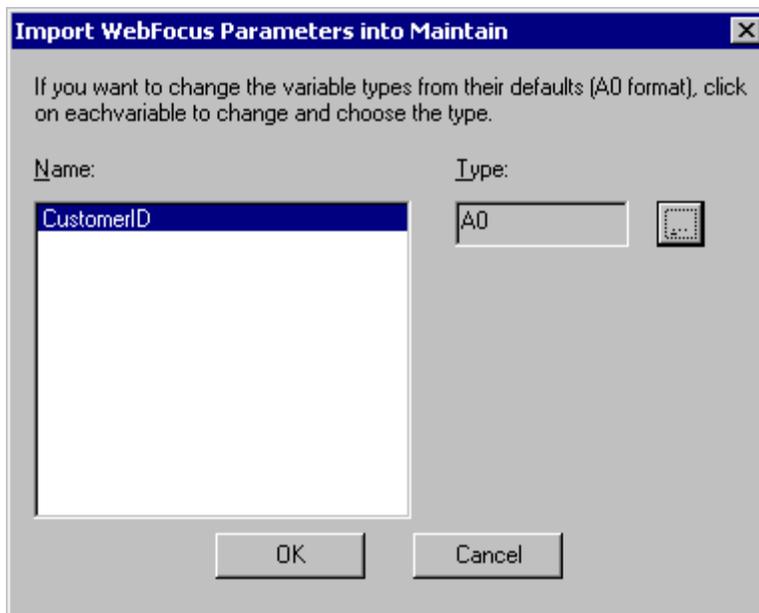
For example, if you use a variable named CustomerNo in a Maintain procedure to determine which record is pulled from a database of customers, but you use CustomerID as the parameter name in the WebFOCUS report that launches the Maintain procedure, it will not work.

You can, however, manually change the variable name in the Maintain procedure to make them match up.

- You cannot import parameters from a WebFOCUS report that is drilling down to a different Maintain procedure.

## Reference Import WebFOCUS Parameters Into Maintain Dialog Box

When you are importing the parameters from a WebFOCUS report into a Maintain procedure, you use the Import WebFOCUS Parameters Into Maintain dialog box to change the formats of the parameters.



This dialog box has the following fields:

**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 2-23.

## 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 like 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; that is, 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.

**Caution:** When you import 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**

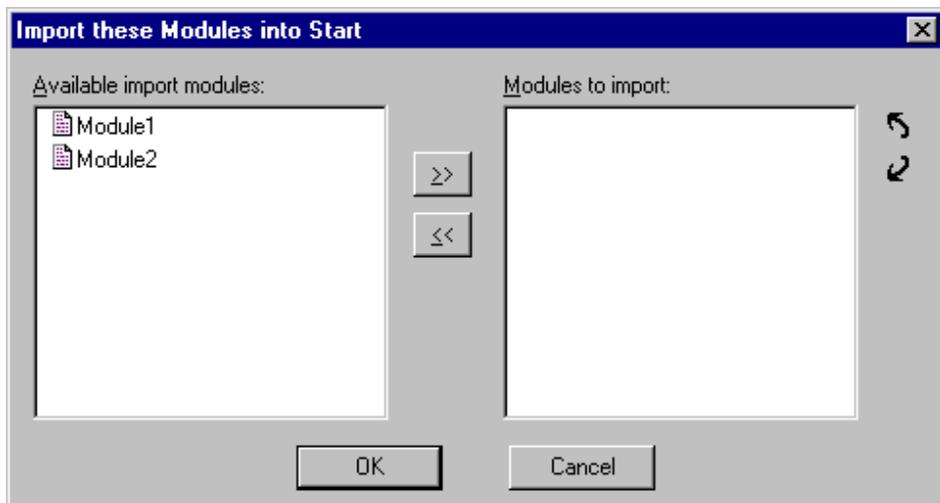
1. Right-click the procedure.
2. In the shortcut menu, click *Import modules...*  
The Import these Modules into Procedure dialog box opens.
3. Select the import modules you want to use in the *Available import modules* list and click the  button.
4. To remove import modules you no longer need, select them in the *Modules to import* list and click the  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 specified an import module for the procedure, WebFOCUS Maintain creates an Import Modules folder under the procedure and lists the import modules you specified inside.

**Tip:** You can also drag an import module to the procedure you want to use it in.

**Reference Import These Modules Into Procedure Dialog Box**

You specify the import modules to use in your procedure with the Import these Modules into Procedure dialog box.



This dialog box contains the following fields:

**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 will use.



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



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

---

---

## CHAPTER 3

# Using the Form Editor

### Topics:

- Layout of the Form Editor
- Using the Controls Palette
- Using the Property Sheet
- Using Drawing Aids

The Form Editor allows 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.

The Form Editor is where you design 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 Chapter 6, *Developing and Using Controls*.

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.

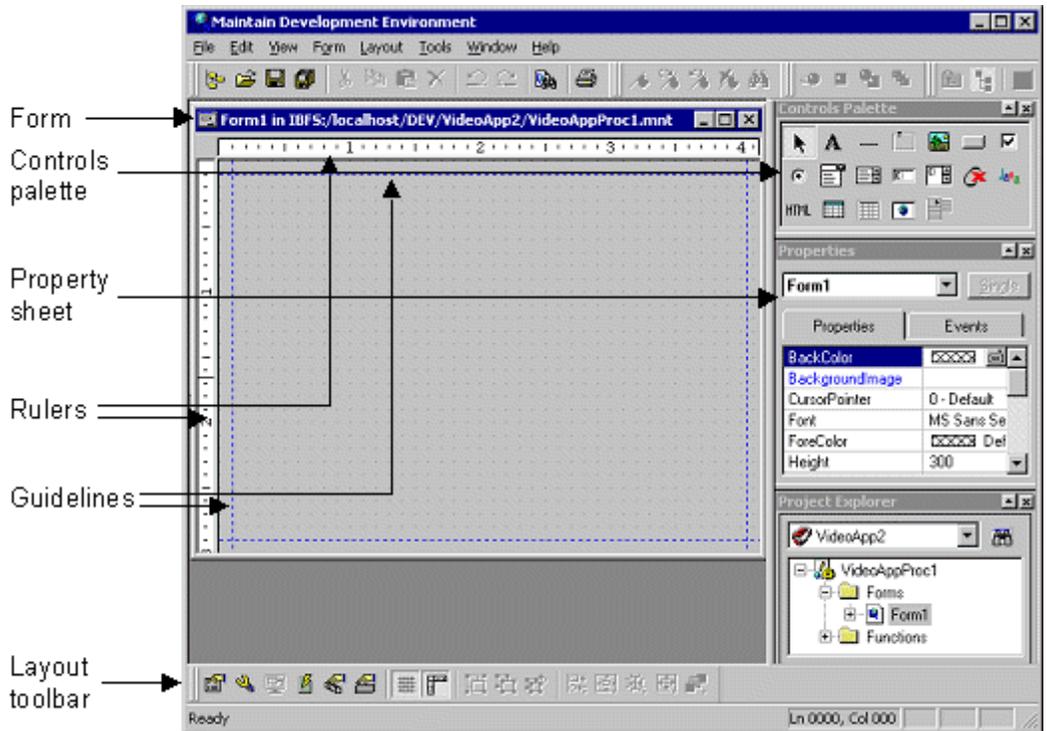
Most properties can be changed dynamically at run time.

For more information on properties, see *Form and Control Properties Reference* in *WebFOCUS Maintain Language Reference*.

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 Chapter 5, *Defining Events and Event Handlers*.

## Layout of the Form Editor



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 your 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

Help you position controls on a form by displaying inch markings at the top and left of the form.

### Guidelines

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

### Layout toolbar

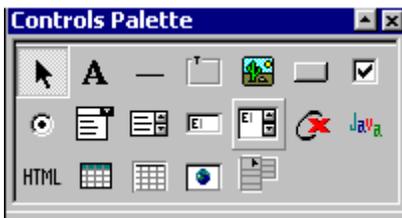
Contains commands that manipulate forms and controls on a form.

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

## Using the Controls Palette

---

The Controls palette contains the controls that you can place on your form.



The first control, the Select control, is the only one that does not place something on your form. Instead, you 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 need to 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).

Chapter 6, *Developing and Using Controls* contains specific information about each control.

## 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 easily 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.

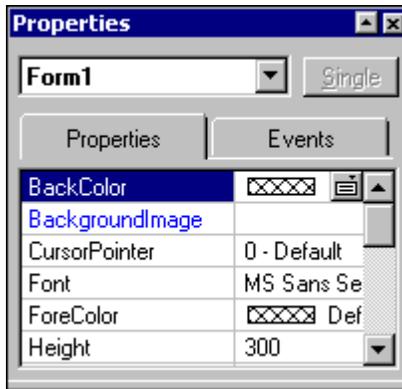
The properties determine what your forms and controls look like and how they behave at run time. Some properties can be bound to data. These properties show up in blue in the property sheet.

You can also set many properties dynamically: that is, 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, you would place the text on the form and set its Visible property to No. When the 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* in Chapter 4, *Developing and Using Forms*, *How to Set the Value of a Control Property* in Chapter 6, *Developing and Using Controls*, and *How to Set the Value of a Control Color Property* in Chapter 6, *Developing and Using Controls*.

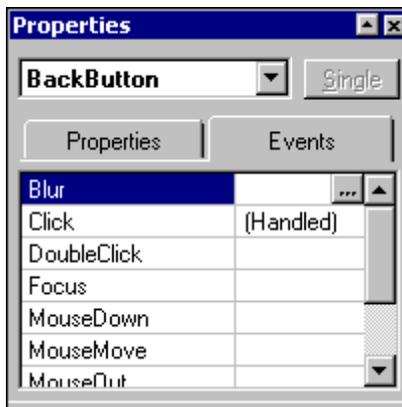
**Note:** You cannot drag properties that cannot be set at run time.

For a description of each property, see *Form and Control Properties Reference* in *WebFOCUS Maintain Language Reference*.



The Events tab in the property sheet contains a list of events for the selected control or form. (*Events* are changes in a form's or controls state that generate 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.

For more information on events, see Chapter 5, *Defining Events and Event Handlers*.



### **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, then click anywhere in the form outside of a control.

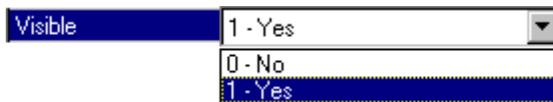
or

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).



For some properties, you must select the value from a drop-down list. Click the  box to the right of the property to see a list of values and select one.



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  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, then 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 (you may need to scroll down in the list to find the one you want) by clicking it. You cannot dynamically set bindable properties (the ones 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, that 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* in *WebFOCUS Maintain Language Reference* 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* in Chapter 6, *Developing and Using Controls*.

### **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 controls around the gridline closest to the center of the control.
5. If you want these settings to apply to this form only, select *Apply to this form only*.

### **Procedure** How to Display Rulers

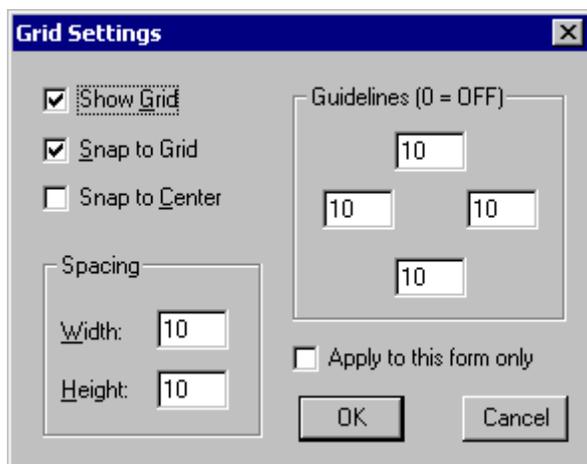
1. Make sure that the form you want to display rulers in is the active window.
2. In the View menu, click *Rulers*.

or

Click the *Toggle rulers*  button on the Layout toolbar.

### **Reference** Grid Settings Dialog Box

You use the Grid Settings dialog box to determine settings for your grid.



#### **Show Grid**

Displays the grid on the form.

#### **Snap to Grid**

Causes controls to snap to the gridlines.

#### **Snap to Center**

Causes controls to snap to the center of the grid.

**Spacing**

Determines the distance between gridlines. The default is 10 pixels.

**Guidelines**

Determines how much blank space to leave at the top, bottom, right, and left sides of a form. You will see blue lines on your form representing where this border begins. You cannot place controls outside of these blue lines.

**Apply to this form only**

Applies these settings to this form only, leaving the settings for other forms unchanged.

---

---

## CHAPTER 4

# Developing and Using Forms

### Topics:

- Before You Begin Designing Forms
- Creating and Managing Forms
- Changing Form Properties
- Dynamically Manipulating Forms at Run Time
- Using a Driver Procedure

A form is an interface that enables an end user to interact with an application. You can create an application using multiple forms.

This topic describes how to design a form, how to manage a form, and how to manipulate a form.

## Before You Begin Designing Forms

---

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

- *Try to adopt the end user's point of view.* 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 will ensure that all of your forms look alike, and will make 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 will ensure that end users will get used to the application more quickly.
- *Give your application the same "look--and--feel" as other applications or web pages at your site.* Once again, this will help end users learn your application more easily and will give your application more credibility.
- *Determine the "flow" of all of the forms in your application and implement a driver procedure.* By this, we mean determine what forms you will have and which ones call each other. Then implement the driver procedure discussed in *Using a Driver Procedure* on page 4-11. Using the driver procedure reduces the likelihood of seeing one of the most common WebFOCUS Maintain application errors—the dreaded "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.

## Creating and Managing Forms

---

You create and manage forms using the Project Explorer. From here, you can:

- Create forms.
- Open forms for editing.
- Export and import forms.
- Rename forms.

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 and the new name will automatically be reflected in the WINFORM SHOW statement.

**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 which procedure to place your forms in.

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.

Once 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

Sometimes you might 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.

## Changing Form Properties

---

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

*Do you want to change the title of the form that is displayed to the end user?*

Use the Title property.

*Do you want to change the size or location of the form (not currently used because of how WebFOCUS Maintain deploys applications)?*

Change the Height, Left, Top, and Width properties.

*Do you want to add or remove standard window components to or from your form (not currently used because of how WebFOCUS Maintain deploys applications)?*

Use the MaximizeBox, MinimizeBox, ScrollBars, ScrollHeight, ScrollWidth, and Sizeable properties.

*Do you want to change the color of the form?*

Use the BackColor property.

*Do you want to add an image to the background of the form?*

Use the BackgroundImage property. See also *Using Images* in Chapter 6, *Developing and Using Controls*.

*Do you want to change the font of all controls in the form?*

Use the Font property to change the typeface, style, and size of the text. Use the ForeColor property to change the color of the text.

*Do you want to change what the cursor looks like when it is on top of the form?*

Use the CursorPointer property.

*Do you want to display a tool tip when the cursor is on top of the form?*

Use the ToolTipText property.

*Do 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* in Chapter 6, *Developing and Using Controls*.

## **Dynamically Manipulating Forms at Run Time**

---

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

If you are developing with forms, you need to 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 form's connection to the server while continuing to display the form.

This is *non-persistent mode* — 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 will not be 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 will not be 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. You 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 need to 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 will be 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 Chapter 5, *Defining Events and Event Handlers*.)

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. (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 easily 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:

```
WINFORM 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:

```
WINFORM SHOW_INACTIVE formname [ ; ]
```

Control passes to the next command in the procedure, 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—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 will be creating a persistent form. If you use the code that is imbedded in the comment text at the top of each new form, WINFORM SHOW\_AND\_EXIT, you will be creating a non-persistent form.

Forms in Maintain applications that were built in Version 4 Release 2.1 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 user's browser. Once 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 WebFOCUS Server agent is no longer dedicated to that user, and is free 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 and Hiding Forms at Run Time**

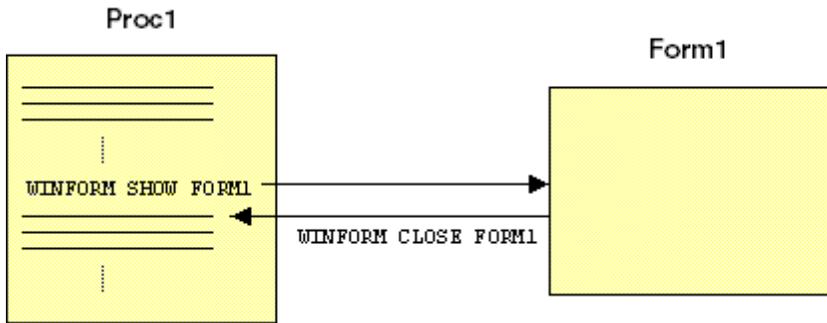
Maintain has several commands for closing or hiding forms at run time:

- The WINFORM CLOSE *form* and SELF.WINCLOSE commands close a form and return control to the command following the command that opened the form.

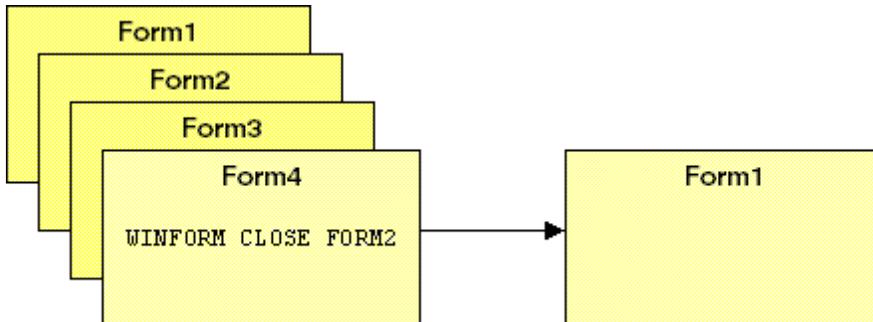
**Note:** You can only issue SELF.WINCLOSE in the Event Handler editor for a form or one of its controls.

- 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. **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 will close Form2, Form3, and Form4, and return to Form1.



WINFORM CLOSEing a form 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 will close the specified form and the chain of forms that may have been opened from this form. 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 will 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 easily 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:

```
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

```
WINFORM GET form.property INTO variable [;]
```

where:

*form*

Is the name of the form.

*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

```
[WINFORM SHOW_INACTIVE form;]  
WINFORM SET form.property TO setting [;]
```

where:

WINFORM SHOW\_INACTIVE

Is required if the form is not the active form.

*form*

Is the name of the form.

*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* in *WebFOCUS Maintain Language Reference*.

**Tip:** You can easily 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* in Chapter 6, *Developing and Using Controls*.

## **Syntax**      **How to Reset a Form**

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

```
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 4-6.

## **Syntax**      **How to Close the Application**

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

```
SELF.WINEXIT ( ) ;
```

### **Note:**

- The SELF.WINEXIT syntax only works in the Event Handler editor for a form or one of its controls.
- WebFOCUS Maintain will ignore any commands after this one.

**Tip:** You can easily generate the SELF.WINEXIT syntax in the Event Handler editor by clicking the *Close Form*  button.

## **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, so it is easy to open more and more forms with impunity 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 and your host platform can’t possibly handle holding that many in memory.

To avoid this problem, we recommend having only one form open at a time—that is, when you open one form, you close the previous one. However, placing the following syntax in your procedure will *not* work:

```
WINFORM CLOSE FORM1
WINFORM SHOW FORM2
```

Why not? 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.

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

The theory behind the driver procedure is simple. Have a variable (say 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 "or some other ending string.

### **Example** Coding a Driver Procedure

Here is an example of some of a driver procedure:

```
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
      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:

```
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. In this way, we will only ever have one form 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. Let's take a 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
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 will know 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 will not have a Stack Overflow problem in your application because you will only have one form open at any given time.

By the way, you can tell that it is very easy to write one of these driver loops if you diagram your Form flow—that's why it is so important to do that.

---

---

## CHAPTER 5

# Defining Events and Event Handlers

### Topics:

- 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

Once you define what your form looks like, you need to define the *event handlers*—what occurs 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.

## 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. (See *How to Open the Event Handler Editor* on page 5-2.)
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:
  - Invoke a Maintain function.
  - Invoke a JavaScript function.
  - Invoke a VBScript function.
  - Go to a Web link.
  - Close the form (the *Maintain function* button must be selected). See *Closing and Hiding Forms at Run Time* in Chapter 4, *Developing and Using Forms*.
  - Close the application (the *Maintain function* button must be selected). See *Exiting Your Application at Run Time* in Chapter 4, *Developing and Using Forms*.

### **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 and, in the shortcut menu, click *Edit event handlers...*

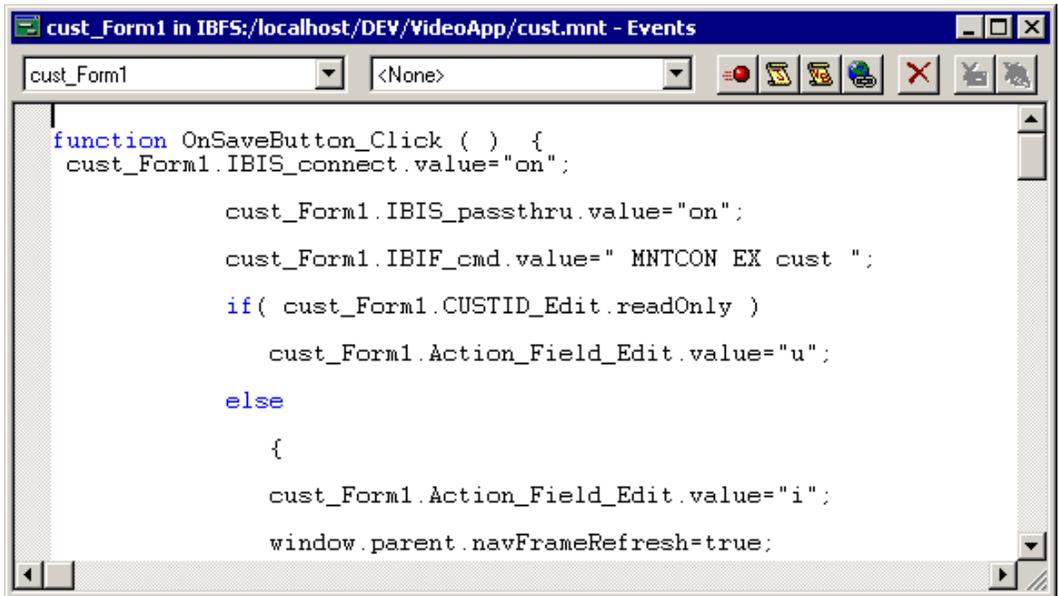
(The control will be selected in the list of controls in the Event Handler editor.)

**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.



The screenshot shows a window titled "cust\_Form1 in IBFS:/localhost/DEV/VideoApp/cust.mnt - Events". The window contains a text editor with the following JavaScript code:

```
function OnSaveButton_Click ( ) {
  cust_Form1.IBIS_connect.value="on";

  cust_Form1.IBIS_passthru.value="on";
  cust_Form1.IBIF_cmd.value=" MNTCON EX cust ";
  if( cust_Form1.CUSTID_Edit.readOnly )
    cust_Form1.Action_Field_Edit.value="u";
  else
  {
    cust_Form1.Action_Field_Edit.value="i";
    window.parent.navFrameRefresh=true;
  }
}
```

This editor has the following:

### 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 5-5.



If the *Maintain* function button is selected, it indicates that your event handler will be a Maintain function.



If the *JavaScript* button is selected, it indicates that your event handler will be a JavaScript function.



If the *VBScript* button is selected, it indicates that your event handler will be a VBScript function.



If the *Web link* button is selected, it indicates that your event handler will be a Web link.



Deletes code for the existing event handler.



If you selected the *Maintain function* button, clicking this button inserts the code for closing your form (`SELF.WINCLOSE();`).

**Caution:** 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 selected the *Maintain function* button, clicking this button inserts the code for closing your application (`self.WinExit();`).

**Caution:** 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 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.

### 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.

### Close

Occurs when 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, 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, 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 down. 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.

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

**MouseOut**

Occurs when an end user moves the mouse pointer off of 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.

**OnCanViewMove, OnCBDClicked, OnCBLClicked, OnCBRClicked, OnCellChange, OnCellChanged, OnCharDown, OnColChange, OnColSized, OnEditFinish, OnEditStart, OnHitBottom, OnKeyDown, OnLClicked, OnRClicked, OnRowChange, OnRowSized, OnRowSizing, OnSHDClicked, OnSHLClicked, OnSHRClicked, OnTHDClicked, OnTHLClicked, OnTHRClicked**

Occurs only in the Grid control. For more information, see *Events Available in a Read/Write Grid* in Chapter 6, *Developing and Using Controls*.

### **Open**

Occurs when a form is opened or shown.

**Note:** You can only use a Maintain function as an event handler for this event, not 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**

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 could 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.

**Caution:** You cannot place CALL or EXEC statements in the Event Handler editor. You must place them in a Maintain function, and then perform the function from the Event Handler editor.

### **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 will execute the function.

## Using Script Functions as Event Handlers

---

One of the components in a WebFOCUS Maintain project is a *script*. A script contains standalone functions, written in either *JavaScript* or *VBScript*, that are embedded in your form for execution on the client.

Why use scripts? JavaScript and VBScript functions can perform *local* validation—that is, 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 the 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 don't 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.

For more information, see the Netscape home page at <http://home.netscape.com/>.

*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.

For more information on VBScript, check out <http://www.microsoft.com/vbscript/>.

This help system document will show how to use several JavaScript functions; however, it is not a complete guide to JavaScript.

**Tip:** Unlike Maintain, JavaScript and VBScript are *very* 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 5-10.
- Use the Explorer to create JavaScript or VBScript libraries and enter the code for your functions there. Then 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. For more information, see *How to Assign a JavaScript or VBScript Function in a Script Library to an Event* on page 5-10 and *Managing Script Libraries* on page 5-17.

### **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*  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 on script libraries, see *Managing Script Libraries* on page 5-17).

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*  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( );
```
6. In the Project Explorer, drag the script library that contains the function to the form in which you are using it. (This ensures that WebFOCUS Maintain includes the script only in the page that is using it.)

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 will need to 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 5-11, *How to Use the JavaScript Confirm Function* on page 5-11, and *How to Use the JavaScript Prompt Function* on page 5-12.

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 will need to 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 they executed it from. If the validation test succeeds, the function needs to return control to the WebFOCUS Maintain application, usually to a Maintain function. It does this 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 5-12.

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. For more information, see *How to Determine Row Number, Column Number, or Cell Value When an HTML Table is Clicked via a Script* on page 5-13.

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). For more information, see the *Using Functions* manual.

**Tip:** Unlike Maintain, JavaScript and VBScript are *very* 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

```
alert("text");
```

where:

*text*

Is informational text that is displayed in the pop-up window. You can enter a variable name here as well (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

```
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 as well (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

```
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 as well (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

```
IWCTrigger("functionname" [, "parameter"]
```

where:

*functionname*

Is the name of the Maintain function you are calling.

**Tip:** Scripts are *very* 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

```
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 you are using the script library in.

**Tip:** 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.

**Syntax**

**How to Determine Row Number, Column Number, or Cell Value When an HTML Table is Clicked via 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:

```
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:

```
document.formname.tablename_ClickColumn.value
```

If you want to determine the contents (text) of the cell, use:

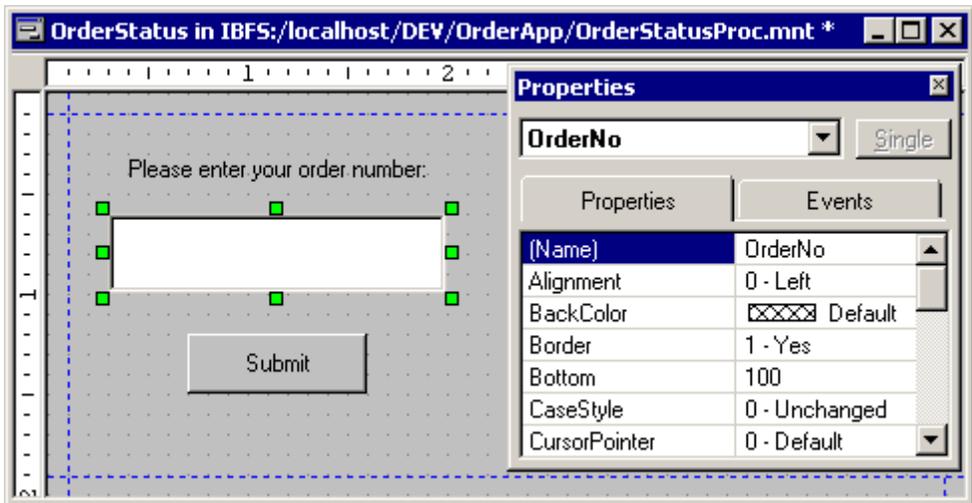
```
document.formname.tablename_Value.value
```

**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 Submit button's Click event 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.



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:

```
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++)
        {
4.     var ch=fldvalue.substring(i,i+1)
        if (ch<"0" || "9"<ch)
        {
5.         if (ch != ".")
            {
                okay=0
                alert("Please enter only numeric data!");
                break
            }
        }
    }
6.   if (okay==1)
        IWCTrigger("DoProcess")
    }
}
```

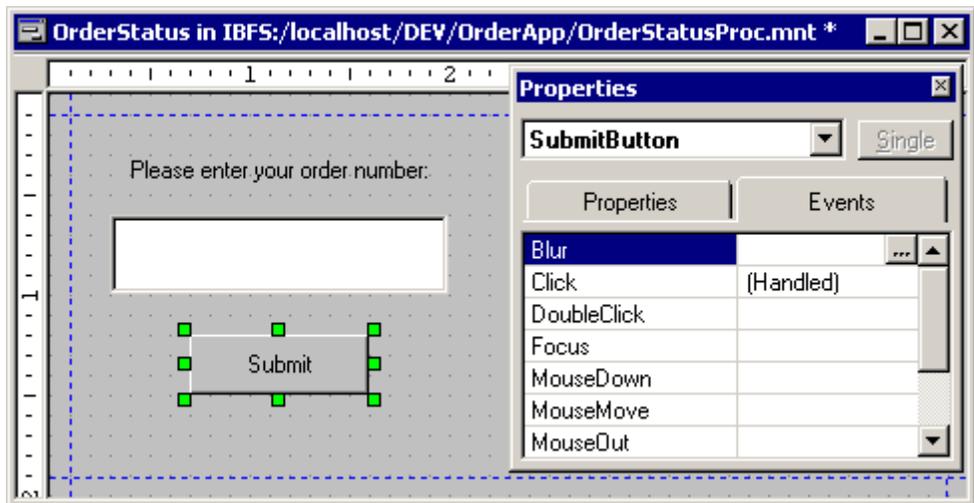
1. This line of code sets **fldvalue** to be the **value** of **OrderNo** on **OrderStatus** on this HTML **document**.
2. The first **if** test checks to see if **fldvalue** is blank. If so, it pops up a dialog box saying, "Please enter some data!"
3. If **fldvalue** is not blank, then test\_it enters 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 (via IWCTrigger).

With the function complete, you now need to assign it to the Submit button in place of DoProcess. See *Executing a JavaScript Function From a Button* on page 5-16.

### Example Executing a JavaScript Function From a Button

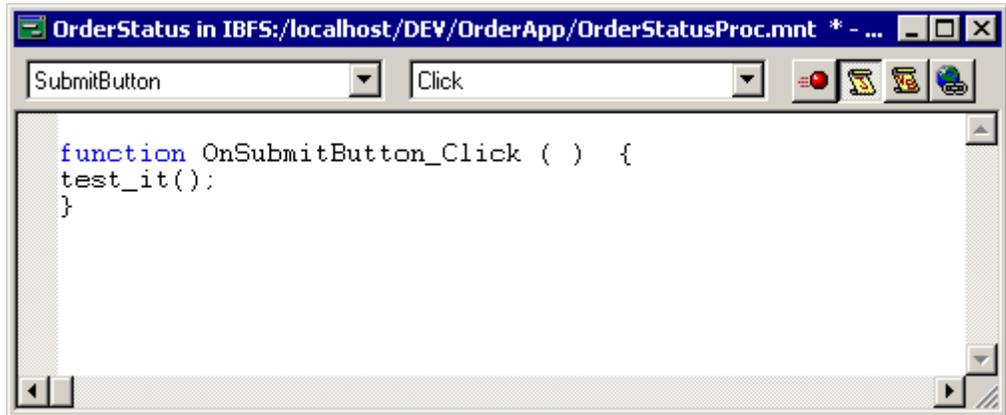
In *Validating End User Input With JavaScript* on page 5-13, you started with an application that prompts an end user to enter a numerical value (in the OrderNo edit field) and click a *Submit* button. The application then executes a Maintain function called DoProcess. (The Submit button's Click event is assigned to a Maintain function called DoProcess.)

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 now need to assign it to the Submit button in place of DoProcess.



1. In the Project Explorer, drag Script1 to OrderStatus. This ensures that WebFOCUS Maintain includes this script only in OrderStatus.
2. Open the Event Handler editor for the form OrderStatus.
3. From the list of controls on the form, select *SubmitButton*.
4. From the list of events that could happen to that control, select *Click*.
5. Click the *JavaScript* button to indicate that clicking *SubmitButton* would execute a JavaScript.

6. Type `test_it();` in the box.



Now, when end users executing this WebFOCUS Maintain application click the *Submit* button, their browsers execute the `test_it` function.

## 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* in Chapter 1, *Working in the Maintain Development Environment*. (The default extension for JavaScript files is `.js`; the default extension for VBScript files is `.vbs`)

Then, from the Explorer you can:

- Create script libraries (see *How to Create a Project Component* in Chapter 1, *Working in the Maintain Development Environment*).
- Rename script libraries (see *How to Rename a Project or Project Component* in Chapter 1, *Working in the Maintain Development Environment*).
- Remove script libraries from the project (see *How to Remove a Project Component From a Project* in Chapter 1, *Working in the Maintain Development Environment*).
- Delete script libraries (see *How to Delete a Project or Project Component* in Chapter 1, *Working in the Maintain Development Environment*).
- 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. See *Using Script Functions in Your Project* on page 5-9 for more information.

Your next step is to associate your script library with a form. See *How to Associate a Script Library With a Form* on page 5-18.

**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 Project Explorer, select the script library.
2. Drag the script library to the form.

Your last step is to assign the functions you have coded in your script library to events. See *How to Assign a JavaScript or VBScript Function in a Script Library to an Event* on page 5-10.

## 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. (You can double-click the line in the Output window to open the script to the offending line.)

Keep in mind that JavaScript and VBScript are *very* case-sensitive, unlike the Maintain language, 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 Web page, e-mail address, or FTP server (basically anything that can be typed into a browser's Location box). A Web link can also be an HTML file.

Once you create a Web link, you can use it as an event handler in your application. For example, you might define a Web link that maps to a Web page. In the Form Editor, you might assign this Web link to an event that occurs when a user clicks a button. During run time, clicking this button opens the Web page 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* in Chapter 6, *Developing and Using Controls*.

You can also launch a Web link into a frame using the frame control. For more information, see *Using Frames* in Chapter 6, *Developing and Using Controls*.

Here are some ways you might use Web links:

- Create a button on every form that jumps to your company's home page.
- Create a Suggestions button. Clicking this button would enable end users to compose and send e-mail to the staff responsible for administrating the application. (E-mail links require that the end user's browser 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, 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  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 5-20, *How to Use the URL Wizard to Define an FTP Web Link* on page 5-20, *How to Use the URL Wizard to Define a WebFOCUS Report as a Web Link* on page 5-20, and *How to Use the URL Wizard to Define an Email Web Link* on page 5-21.

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.

**Procedure** **How to Use the URL Wizard to Define an HTTP Web Link**

1. In the Link Editor, click the  button.
2. In the URL Wizard, select *http* or *https: - Hypertext Transfer Protocol* and click *Next*.
3. Enter the domain name of your target server.
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. If the object is a CGI or ISAPI program, select *This object is a Common Gateway Interface (CGI) or Internet Server Application Programming Interface (ISAPI) program*. Otherwise, click *Finish*.
8. Click *Next*.
9. If your CGI or ISAPI program has any parameters whose value you must supply at run time, define them and supply values.
10. Click *Finish*.

**Procedure** **How to Use the URL Wizard to Define an FTP Web Link**

1. In the Link Editor, click the  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  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. (Notice that the URL Wizard appends the text */cgi-bin/ibi\_cgi/ibiweb.exe* after the domain name. This points to the WebFOCUS executable.)  
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  button.
2. Define the recipients who will receive your e-mail message and click *Next*.
3. If you wish, define the CC recipients who will receive your e-mail message and click *Next*.
4. 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  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 pre-defined 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. In the Project Explorer, right-click the Web link you want to edit.
2. In the shortcut menu, click *Edit...*
3. Make any necessary changes in the Link Editor dialog box.
4. Click *OK* when you are done.

**Procedure** **How to Add a Web Link to a Form Quickly**

1. Select the Web link in the Project Explorer.
2. Drag it to the form you want to add it to.

WebFOCUS Maintain creates a text object in the form, with the text of the Web link and the Click event assigned to the link. You can change the text, but the correct URL remains.

**Procedure** **How to View a Web Link**

1. Right-click the Web link you want to view in the Project Explorer.
2. In the shortcut menu, click *View in Browser*.

WebFOCUS Maintain opens your machine's default browser with the Web link's URL.

**Procedure** **How to View the Description of a Web Link**

When you create a Web link, the Description tab enables you to enter a description for that component. To view this description:

1. Select the Web link in the Project Explorer.
2. Press F4.

or

Right-click the Web link, and in the shortcut menu, click *Show description*.

or

In the Object menu, click *Show description*.

or

Click the *Show description*  button in the Miscellaneous toolbar.

**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

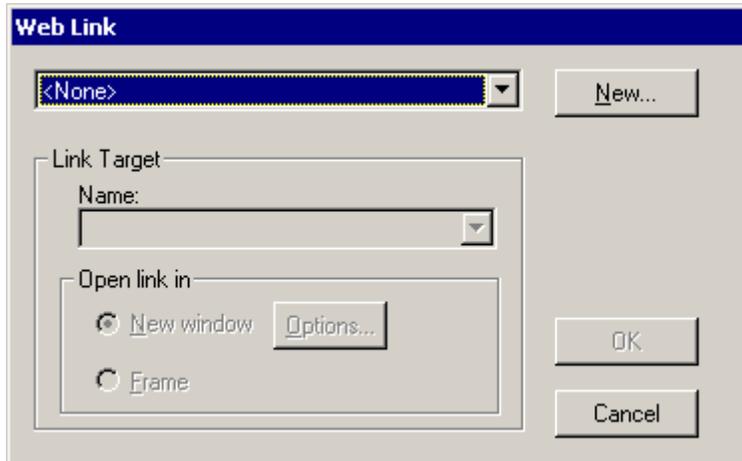
Press the Delete key.

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.



This dialog box has the following fields:

### **[list of available Web links]**

Select the name of an existing Web link in the list.

### **New...**

Opens the Link Editor, where you can define a new Web link.

### **Link Target Name**

Specifies the name of a target you can launch your Web link in. 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**

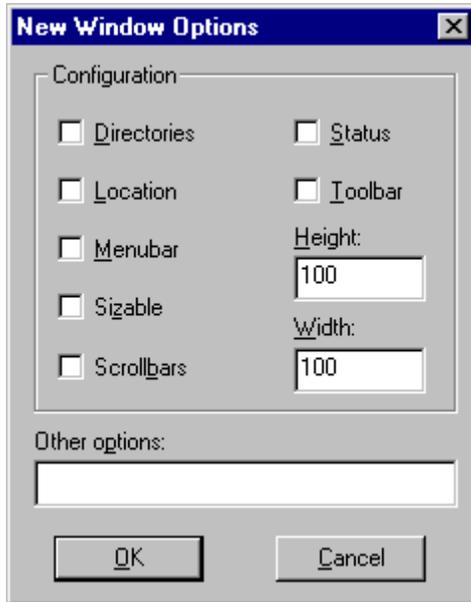
Select *New window* to open your Web link in a new window; select *Frame* to open your Web link in a frame on the current form.

### **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 determines the appearance of the new browser window that your Web link appears in. 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**

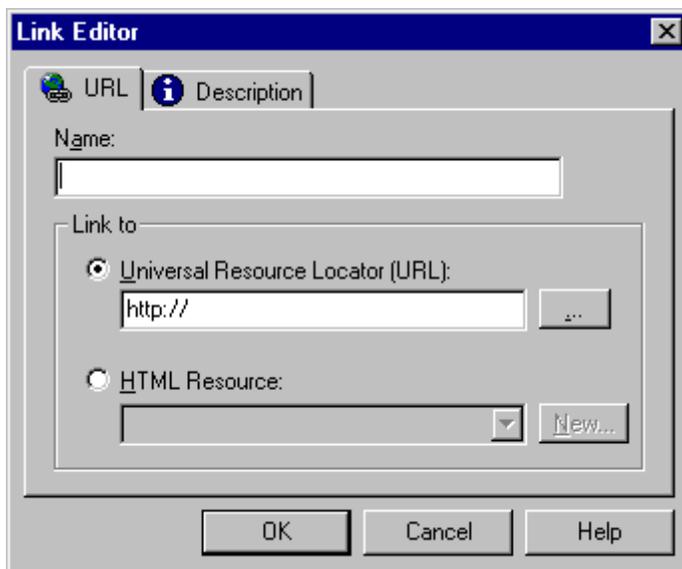
Determines the width of the browser window.

**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.



This dialog box has the following fields:

**Name**

Type the name that you want to refer to this link by.

**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.

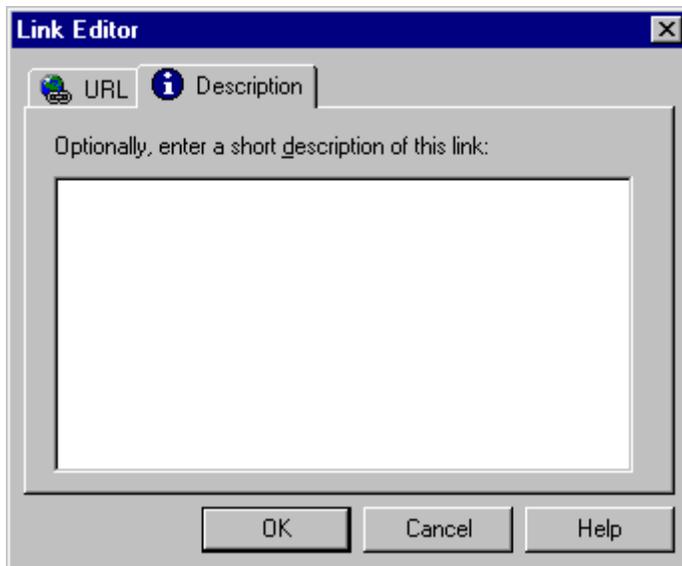
### New...

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**

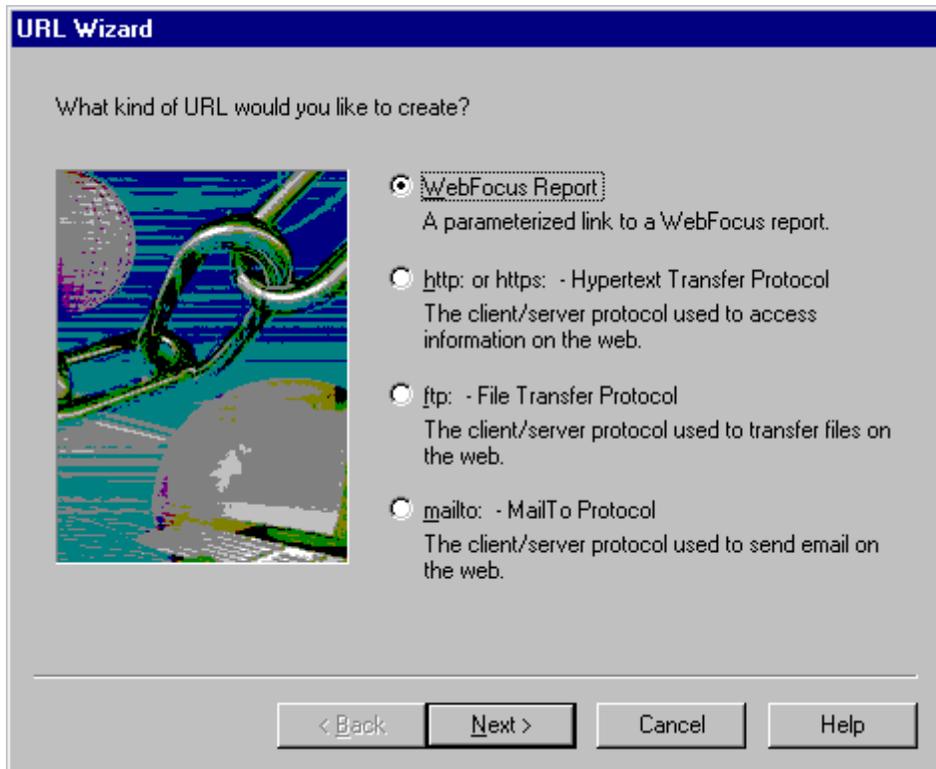
When you click the *Description* tab in the Link Editor, WebFOCUS Maintain opens a dialog box where you can enter a description for your Web link.

You can easily view the description of a Web link in the Project Explorer with the *Show description* command. For more information, see *How to View the Description of a Web Link* on page 5-23.



## 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 easily from here.



**Reference URL Wizard: Specify a Domain Name**

You use this window to specify the domain name of the WebFOCUS Server, Web server, or FTP server to which you are connecting.

Select or enter the domain name for this URL:

Domain name:

Make this a secure hyperlink (https:)  
Hint: The web server must support the Secure Sockets Layer (SSL).

http://

< Back   Next >   Cancel   Help

**Domain name**

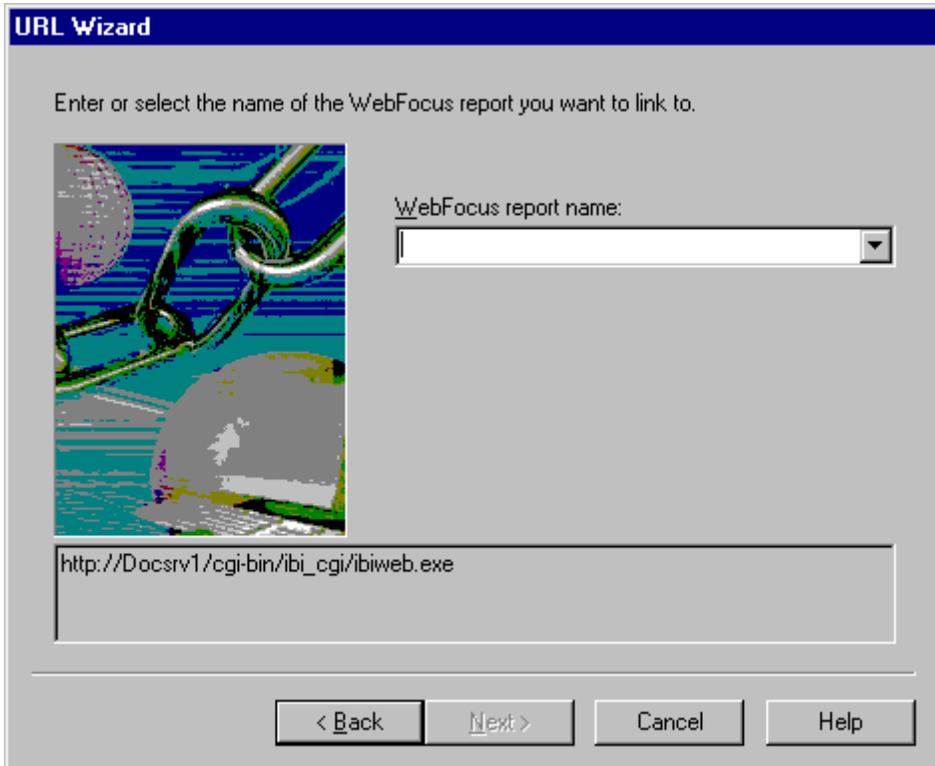
Enter the domain name of your WebFOCUS Server, Web server, or FTP server here.

**Make this a secure hyperlink (https:)**

(For WebFOCUS reports and Hypertext Transfer Protocol only) Select this option if your Web server is secure.

**Reference URL Wizard: Specifying Name of WebFOCUS Report**

You 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 Object**

When you are defining a link to a Web server or FTP server, you have the option of specifying an object.

**URL Wizard**

Enter the object of this URL as a relative path to the domain  
www.informationbuilders.com. The object can be a file, directory, or alias on the web

Object path name:

This object is a Common Gateway Interface (CGI) or Internet Server Application Programming Interface (ISAPI) program

http://www.informationbuilders.com

< Back   Finish   Cancel   Help

### **Object path name**

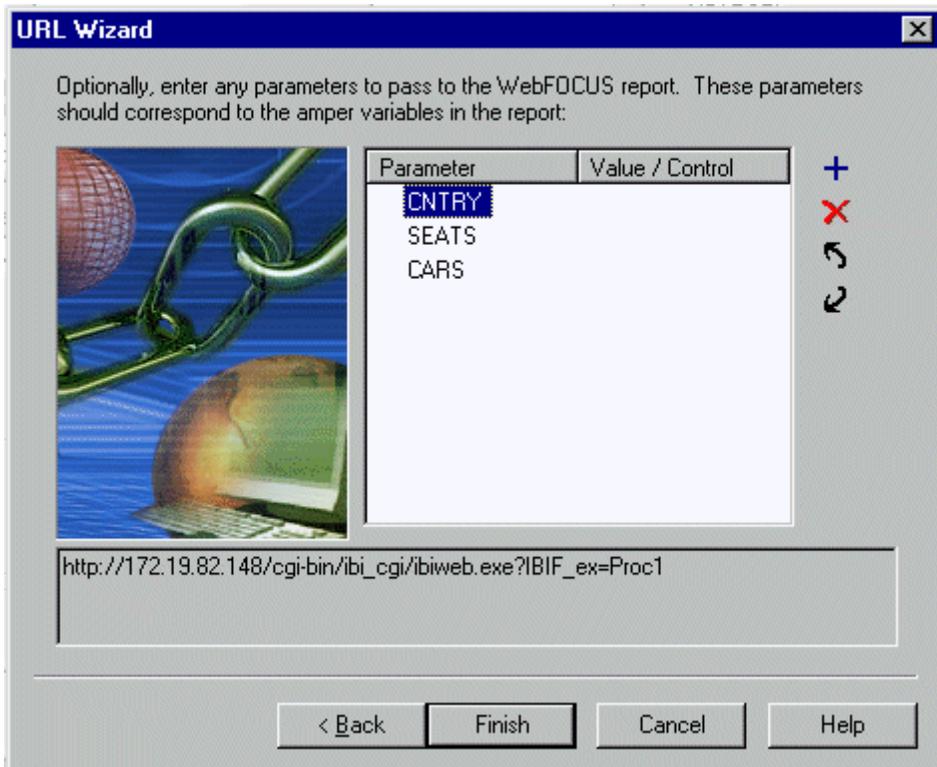
Enter the name of a file, directory, or alias on the Web server or FTP server you are defining a link to. This step is optional.

### **This object is a Common Gateway Interface (CGI) or Internet Server Application Programming Interface (ISAPI) program**

Select this option if the object is one of these types of programs. This changes the Finish button to the Next button so that you can specify values for parameters for these programs, if necessary.

## 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.



### 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:

#### **From a control**

When selected, allows 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.

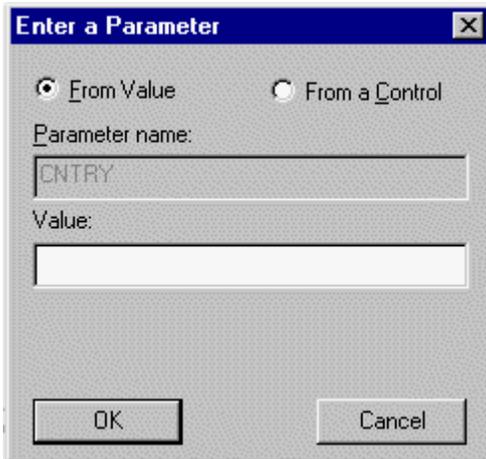
#### **Please select a Form name**

Allows you to specify the name of the form you wish to use to bind your parameter.

#### **Please select a controller**

Allows 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 box is selected:



The screenshot shows a dialog box titled "Enter a Parameter". It has a blue title bar with a close button (X). Inside the dialog, there are two radio buttons: "From Value" (which is selected) and "From a Control". Below the radio buttons, there are two text input fields. The first is labeled "Parameter name:" and contains the text "ENTRY". The second is labeled "Value:" and is currently empty. At the bottom of the dialog, there are two buttons: "OK" and "Cancel".

**From Value**

When selected, allows you to manually specify a value for a parameter.

**Parameter name**

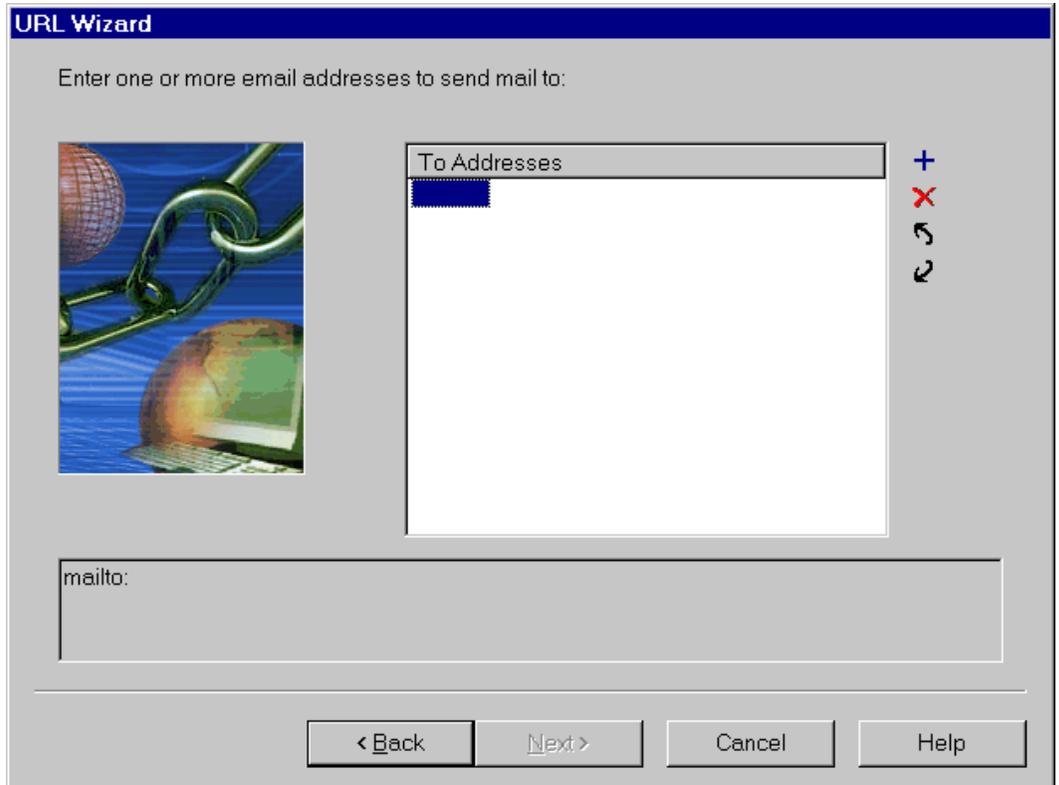
Displays the name of the parameter. If you are adding a parameter, you can name the new parameter here.

**Value**

Allows you to specify the value you wish to bind to the parameter.

## Reference **URL Wizard: Specifying E-mail Addresses**

If you are defining an e-mail Web link, you use this window to define the recipients and CC recipients of your e-mail message.



### To Addresses

Lists the e-mail addresses to which this message will be sent. To edit an address, double-click it here.



Opens the Email Address dialog box, where you can enter a new address.



Deletes the selected address.



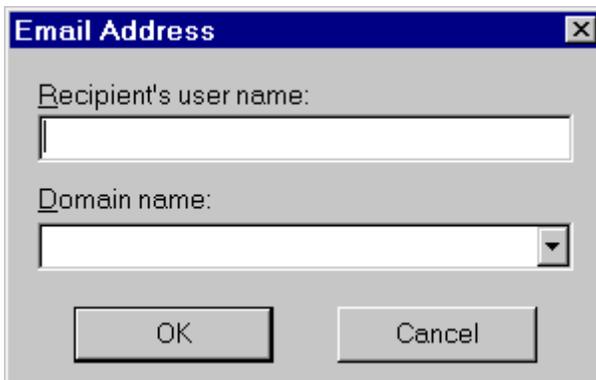
Moves the selected address up in the list.



Moves the selected address down in the list.

### **Reference E-mail Address Dialog Box**

You use the Email Address dialog box to enter a new address.



#### **Recipient's user name**

Enter the user name of your e-mail recipient here.

#### **Domain name**

Enter the domain of your e-mail 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 recipient's user name to the domain name, separated by the @ symbol.

### Reference **URL Wizard: Entering an E-mail Message Subject and Body**

You use this window to enter an optional subject and body for your e-mail message. The end user has the option of changing these fields at run time.

URL Wizard

Optionally, enter a subject and body for this email.

Subject

Body:

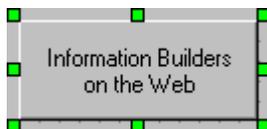
mailto:askinfo@ibi.com

< Back   Finish   Cancel   Help

### Example **Creating and Using a Link**

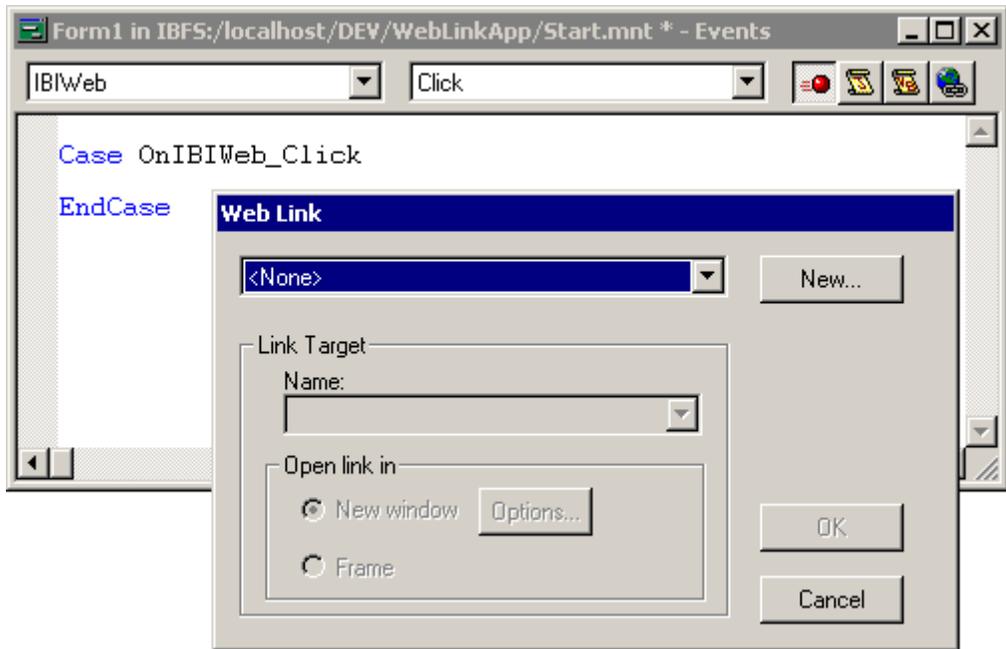
Suppose you decide to include a button in a user application that links to the Information Builders home page: [www.informationbuilders.com](http://www.informationbuilders.com).

1. Open the Form Editor and create a button with the name `IBIWeb` and the title Information Builders on the Web.



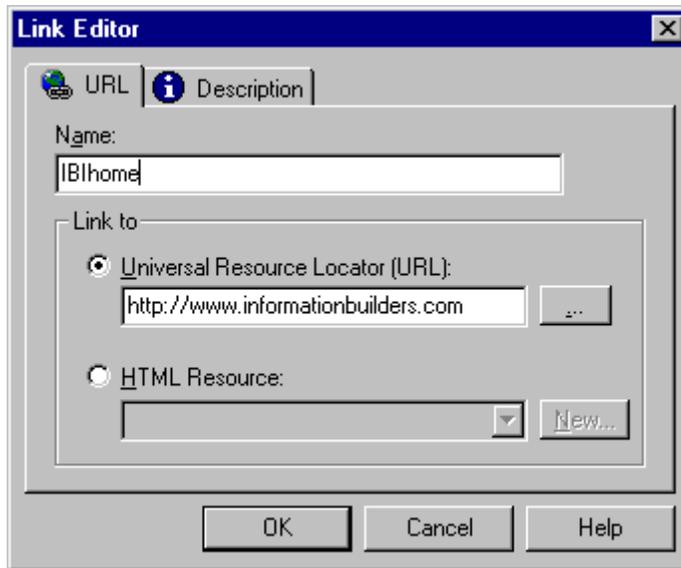
2. Double-click the `IBIWeb` button to open the Event Handler editor.

3. In the Event Handler editor, select the Click event, and then select the Web Link  icon. WebFOCUS Maintain opens the Web Link dialog box.

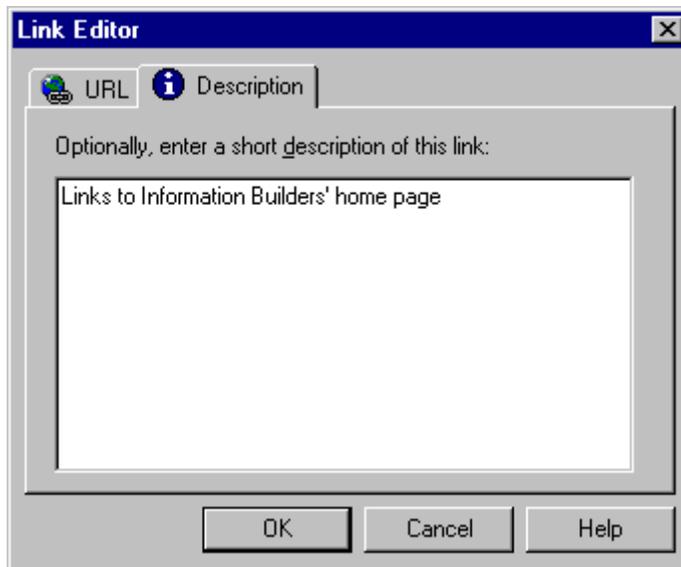


4. Create a web link by clicking New....

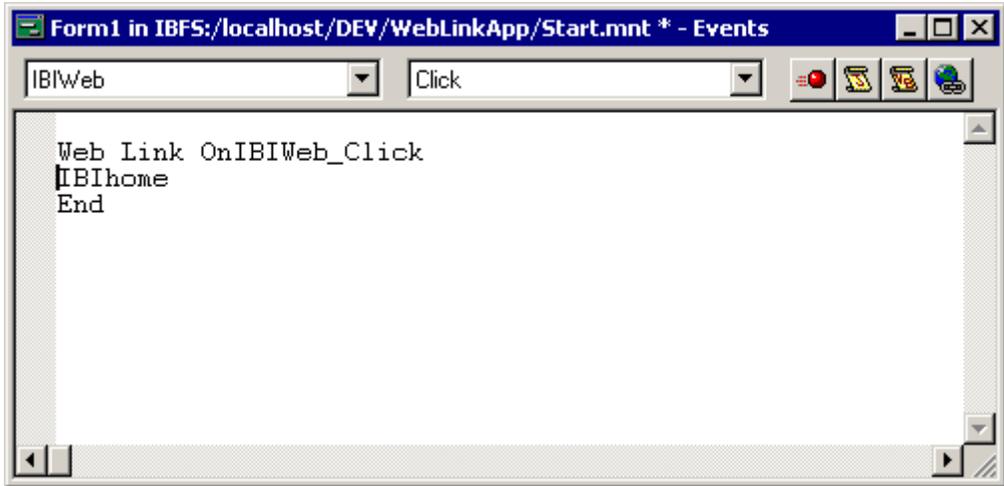
- In the Link Editor dialog box, name your Web link IBIhome and enter the URL `http://www.informationbuilders.com`.



- Enter a description of the Web link in the Description tab.



7. Click *OK* in the Link Editor and the Web Link dialog box.



8. Close the Event Handler editor and save your project.

When you deploy and run the application, clicking *Information Builders on the Web* will open Information Builders' home page.

---

---

## CHAPTER 6

# Developing and Using Controls

### Topics:

- Which Control Should You Use?
- Manipulating Controls
- Using Buttons
- Using Check Boxes
- Using Combo Boxes and List Boxes
- Using Edit Boxes and Multi-Edit Boxes
- Using Frames
- Using Grids
- Using Group Boxes
- Using HTML Objects
- Using HTML Tables
- Using Images
- Using Java Applets
- Using Lines
- Menus
- Using Radio Buttons
- Using Text
- Using ActiveX Controls
- Dynamically Manipulating Controls at Run Time
- Defining Global Attributes

There are many different controls that you can use in your project's 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.

## **Which Control Should You Use?**

---

Which control you use depends on what task you want to perform:

**Do you want to have your end user trigger an action at their convenience?**

Use a button or image.

**Do you want to have your end user turn an option on or off?**

Use a check box.

**Do you want to have your end user select something from a list?**

If the end user can only select one choice, use radio buttons or a combo box.

If the end user can select more than one choice, use a list box or a series of check boxes.

**Do you want to display data to your end user?**

To show static data, use text.

To show data one data field at a time, use an edit box.

To show multiple rows of data from a data source, use an HTML table or a read/write grid.

To show something formatted as HTML, use an HTML object.

To display a Web link in a frame on a form, use a frame.

**Do you want to display a graphic?**

If you want your end user to be able to click the graphic, use an image.

If you want to tile your graphic behind other controls, use a background image on the form.

**Do you want to have your end user edit or provide data?**

Use an edit box.

If a piece of data is more than one line long, use a multi-edit box.

**Do you want to provide your end user with a menu option?**

Use a menu and sub menus.

**Do you want to organize information on your form visually?**

To separate a group of controls from other controls on the form and to associate them with each other, use a group box.

To separate controls from each other visually, use a line.

## Manipulating Controls

---

The WebFOCUS Maintain Form Editor provides many ways for you to manipulate the controls on your form.

- You can easily select all the controls on your form and manipulate them as a group.
- You can 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.
- You can use special commands to resize, align, and space controls.
- You can group controls together so that they are treated as a unit.
- You can undo editing mistakes or redo actions.
- If you are designing complicated forms, you can place controls on separate layers.

### 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 the 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 regenerating the control from scratch 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). For forms, a deep copy copies the controls, event handlers, and connections to resources.

You perform deep copies with the *Paste* and *Duplicate* commands.

- A shallow copy copies appearance only.

You perform shallow copies with 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 (includes appearance, data bindings, events, and connections to resources) in the same form slightly offset from the original control.

**Tip:** You can also hold down the Ctrl key and drag the control to duplicate it.

## 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, or 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 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 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 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.
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 to end users if the controls are neatly aligned. WebFOCUS Maintain provides many alignment tools to help you 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.

You can also center controls in the form either horizontally or vertically.

**Procedure** **How to Align Controls Along Their Left Sides**

1. Select the control whose left side you want the other controls to be aligned to (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**

1. Select the control whose right side you want the other controls to be aligned to (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**

1. Select the control whose top you want the other controls to be aligned to (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**

1. Select the control whose bottom you want the other controls to be aligned to (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**

1. Select the control whose centers you want the other controls to be aligned to (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 whose centers you want the other controls to be aligned to (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.

**Procedure** **How to Center Controls In the Form**

1. Select the controls you want to center in the form.
2. In the Layout menu, click *Center in form*, and then click *Horizontally* or *Vertically* in the submenu.

or

Click the *Center in View*  button in the Layout toolbar.

Then click one of the *Center* buttons 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 WebFOCUS Maintain’s grouping commands to treat several controls as a unit. This means that when you move one of these controls, WebFOCUS Maintain will move all of them together. You can also apply alignment and spacing commands to these controls, and WebFOCUS Maintain will treat 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*.

or

Click the *Ungroup controls*  button in the Layout toolbar.

**Procedure How to Regroup Controls**

- In the Layout menu, click *Regroup*.

or

- Click the *Regroup controls*  button in the Layout toolbar.

**Changing the Order of Controls**

If two controls are taking up the same space on your form, by default, the control that was created most recently is 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 otherwise end users would not be able to click it or type in it.

**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.

## Undoing and Redoing Actions

You can easily 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 Form Editor's layering ability.

Layering works like this: you can create multiple layers and then assign different controls on the form to different layers. You can then hide a layer, thus hiding all the controls on that layer until you make it visible again. You can also lock a layer, thus ensuring that all controls on the layer cannot be changed.

**Note:** Layering is a form editing feature only! You cannot make layers visible and invisible dynamically during run time.

### **Procedure** How to Create a Layer

1. In the Form menu, click *Edit layers...*

or

Click the *Edit layers*  button on the Layout toolbar.

The Layers Sheet dialog box opens.

2. Click *Add*.
3. If you wish, rename your layer by typing over the name in the Name box.
4. Click *OK*.

### **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.

The Layers Sheet dialog box opens.

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.

The Layers Sheet dialog box opens.

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.

The Layers Sheet dialog box opens.

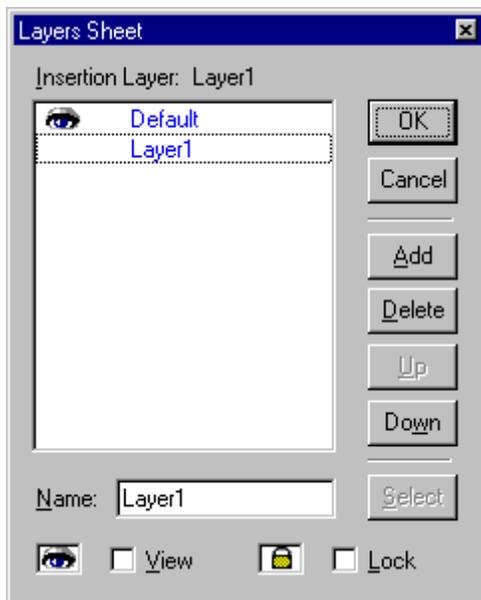
2. Select the layer you want to lock or unlock.
3. Select or clear the Lock box.
4. Click *OK*.

**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.

**Reference** Layers Sheet Dialog Box

You use the Layers dialog box to manipulate the layers in your form. From here, you can create, delete, move, activate, hide, and lock layers.



This dialog box has the following elements:

**List of layers**

Lists the available layers in this form.

**Add**

Creates a new layer on top of the others.

**Delete**

Deletes the selected layer.

**Up**

Moves the selected layer up.

**Down**

Moves the selected layer down.

**Name**

Contains the name of the currently selected layer. If you want to rename a layer, type over the existing name here.

**Select**

Makes the currently selected layer active so that all new controls will be placed on this layer.

**View**

Determines whether you can see the layer.

**Lock**

Determines whether you can edit the controls on a layer or not.

## Using Buttons

---

A button is a control that triggers an action when clicked.

### **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 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.) If you want the text to appear on more than one line, press Shift+Enter between each line.

4. (Optional, but recommended.) Give your button a more meaningful name than `Buttonn`.
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 easily open the Event Handler editor with your button selected by double-clicking the button.

For more information on assigning actions to events, see Chapter 5, *Defining Events and Event Handlers*.

### **Example** Creating a Button to Execute a Maintain Function

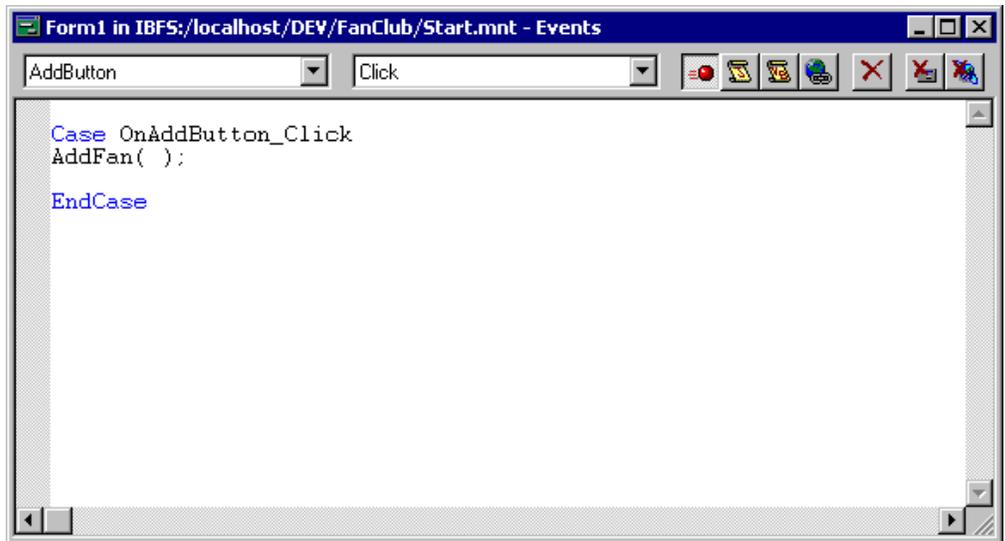
Suppose you want to add a button to your form to execute a Maintain function called `AddFan` (which this example assumes you have already written).

1. In the Form Editor, select the *Button*  control on the Controls palette.
2. Draw a rectangle on your form where you want your button to go at approximately the correct size. (You can move and resize your button later, if necessary.)
3. Type some descriptive, but not too verbose text in your button to describe what action your button will perform. (The text in the button will be already selected.) In this case, the `AddFan` function adds a new fan to the `Fannames` data source, so type the text *Add* into the button.



4. Change the (Name) property from `Button1` to `AddButton`.
5. Double-click the button to open the Event Handler editor. Notice that the `AddButton` control is selected.
6. Select the Click event.
7. Make the Project Explorer the active window, but make sure you can see the Event Handler editor.
8. Click and drag the `AddFan` function from the Project Explorer into the Event Handler editor.

WebFOCUS Maintain automatically generates the code to execute the function.



9. Close the Event Handler editor.

Now, when you run your application, clicking the *Add* button executes the *AddFan* function.

## 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.

### Do you want to change the text displayed in the button?

Change the Text property.

### Do you want to change the name of the button that identifies it to the procedure?

Change the (Name) property.

### Do you want to change the size or location of the button?

Change the Bottom, Left, Right, and Top properties. (You can also move or resize the button directly in the form.)

### Do you want to change the color of the button?

The BackColor property determines what color the button is. The ForeColor property determines the color of the text in the label.

**Do you want to change the text font?**

Use the Font property.

**Do you want to make the button inactive or make it invisible?**

The Enabled property determines whether the button is active or not. (If the button is inactive, it will be grayed out and nothing will happen when the end user clicks it.) The Visible property determines whether the button is visible to the end user.

**Do you want to change what the cursor looks like when it is on top of the button?**

Use the CursorPointer property.

**Do you want to display a tool tip when the cursor is on top of the button?**

Use the ToolTipText property.

**Do 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 6-110.

**Do you want the end user to be able to tab to the button?**

Use the Tabstop property.

**Do you want to make the event handler for clicking the button occur when the end user presses Enter?**

Use the DefaultButton property.

**Do you want to move the button to another layer?**

Use the Layer property. For more information, see *Layering Controls* on page 6-12.

## Using Check Boxes

---

You 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 and type the label you want your check box to have and press Enter.
4. (Optional, but recommended.) Give your check box a more meaningful name than `CheckBoxn`.
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. (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* in Chapter 2, *Developing Procedures* or *How to Create a Data Source Stack Explicitly Using the Stack Editor* in Chapter 2, *Developing Procedures*.)

The variable or stack column should have a value of 1 or 0, if numeric, or a value of TRUE or FALSE, if alphanumeric.

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

Deselects the check box.

1

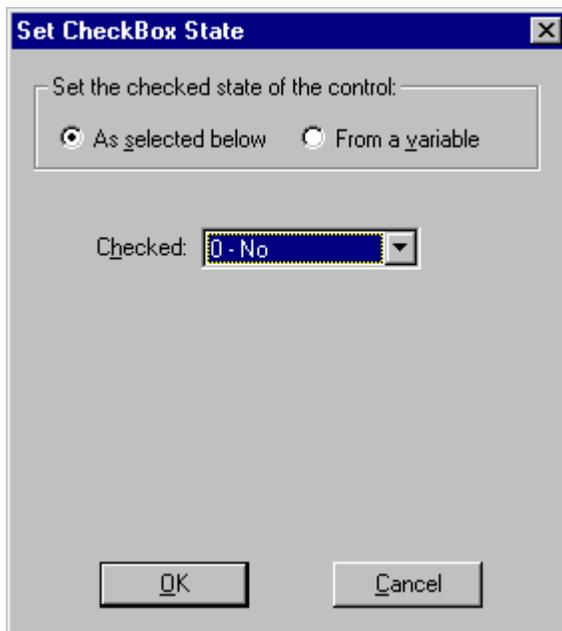
Selects the check box.

**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**

You use the Set Check Box State dialog box 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:

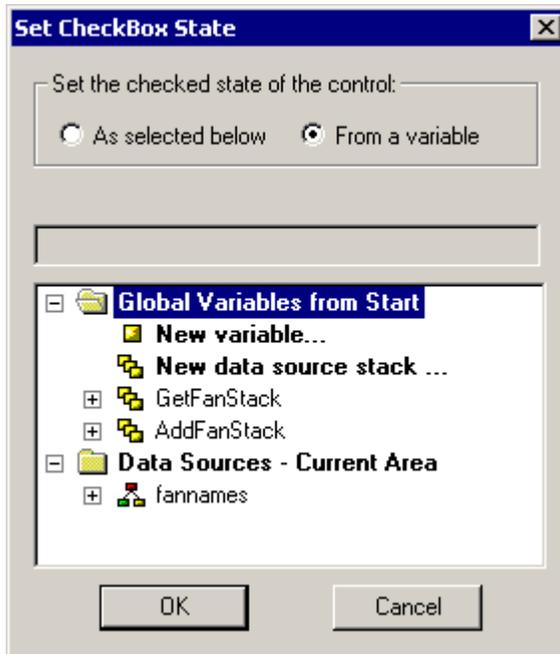
**Set the checked state of the control**

Select *As selected below* to set the initial value directly.

Select *From a variable* to set the value from a variable.

**Checked**

(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...**

(Only if you selected *From a variable*.) Opens the New Variable dialog box, where you can create a variable.

**New data source stack...**

(Only if you selected *From a variable*.) Opens the Stack Editor, where you can create a stack.

**List of data source stacks and variables in your procedure**

(Only if you selected *From a variable*.) 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**

(Only if you selected *From a variable*.) 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.

### **Do you want to change the label that identifies the check box to the end user?**

Change the Text property.

### **Do you want to change the name of the check box that identifies it to the procedure?**

Change the (Name) property.

### **Do you want to change the initial setting for the check box or assign the check box state to a variable?**

Use the Checked property to open the Set Check Box State dialog box.

### **Do you want to change the size or location of the check box?**

Change the Bottom, Left, Right, and Top properties. (You can also move or resize the check box directly in the form.)

### **Do you want to change the color of the check box?**

The BackColor property determines what the background color is. The ForeColor property determines the color of the text in the label.

### **Do you want to change the label font?**

Use the Font property.

### **Do you want to move the label to the other side of the check box?**

Use the TextOnLeft property.

### **Do you want to make the check box inactive or make it invisible?**

The Enabled property determines whether the check box is active or not. (If the check box is inactive, it will be grayed out and nothing will happen when the end user clicks it.) The Visible property determines whether the check box is visible to the end user.

### **Do you want to change what the cursor looks like when it is on top of the check box?**

Use the CursorPointer property.

### **Do you want to display a tool tip when the cursor is on top of the check box?**

Use the ToolTipText property.

### **Do 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 6-110.

**Do you want the end user to be able to tab to the check box?**

Use the Tabstop property.

**Do you want to move the check box to another layer?**

Use the Layer property. For more information, see *Layering Controls* on page 6-12.

## 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. (Optional, but recommended.) Give your combo box or list box a more meaningful name than `ComboBoxn` or `ListBoxn`.
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. You 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 6-24.
  - **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* in Chapter 2, *Developing Procedures*.)

- **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* in Chapter 2, *Developing Procedures*.)

**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. You use this dialog box to bind the end user's 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* or *How to Create a Variable in a Procedure* in Chapter 2, *Developing Procedures*.)

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  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 gray text block in the Text column to open the Enter a List Item dialog box.
- Select the item that you want previous 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 F2.

**Tip:** You can drag text from any application (for example Notepad) on top of a combo box, list box, or group of radio buttons to define the list.

## **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.

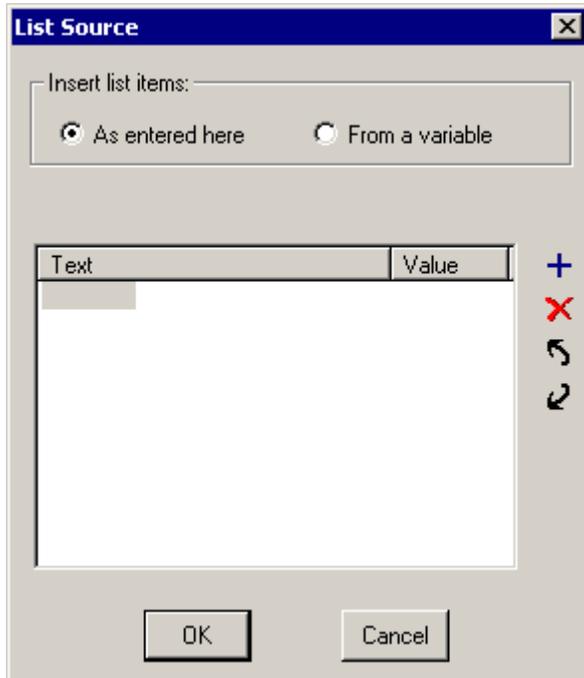
You can also reset the values of all controls in a form to their initial values using the WINFORM RESET command. For more information, see *Manipulating Form Properties* in Chapter 4, *Developing and Using Forms*.

## **Procedure** **How to Trigger an Action When an End User Selects an Item in 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**

You use the List Source dialog box to determine the contents of your list box, radio button, or combo box.



The List Source dialog box contains the following elements:

### **Insert list items**

Select *As entered here* to enter list items manually.

Select *From a variable* to retrieve the list from a data source stack column, variable, or ACCEPT list for a data source field. This enables you to determine at run time what 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.



Enables you to add a selected item to the list of fields by opening the Enter a List Item dialog box.



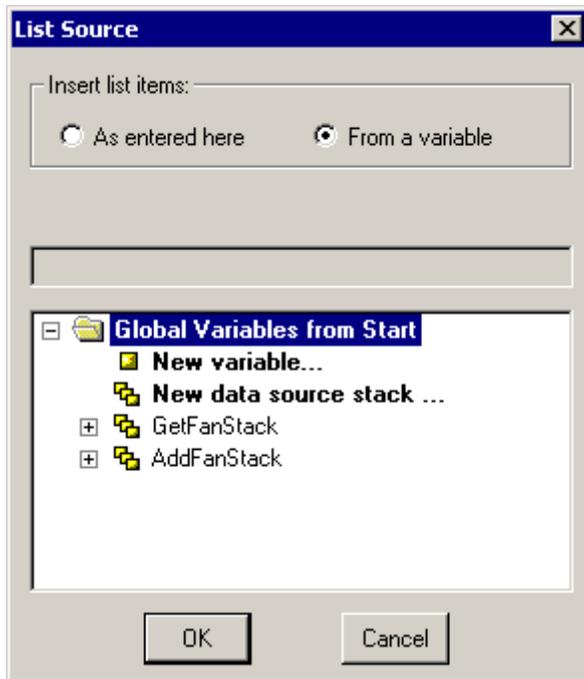
Deletes a selected item from the list of items.



Moves a selected item up in the list of items.



Moves a selected item down in the list of items.



**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**

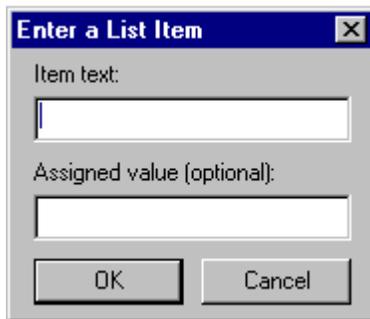
Contains a list of the existing stacks and variables in your procedure. Either select a variable, or expand a data source stack and select a column.

**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 don't have any such fields.)

**Reference Enter a List Item Dialog Box**

You can use the Enter a List Item dialog box to manually enter list items and assign values to the items you enter.



This dialog box contains the following elements:

**Item text**

Contains the list item displayed to the end user.

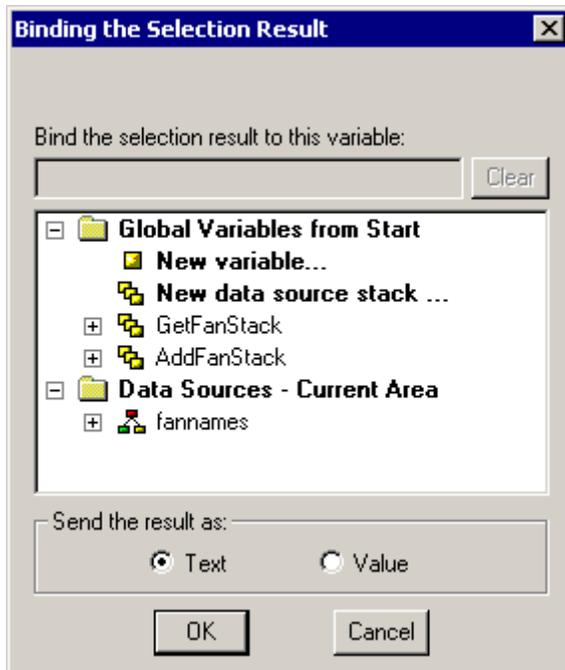
**Assigned value (optional)**

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 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 stack column**

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.

**Do you want to change the contents of the combo box or list box?**

Use the *ListItems* property to open the List Source dialog box.

**Do you want to change the name of the combo box or list box that identifies it to the procedure?**

Change the (Name) property.

**Do 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.

**Do 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.)

**Do you want to change the size or location of the combo box or list box?**

Change the *Bottom*, *Left*, *Right*, and *Top* properties. (You can also move or resize the combo box or list box directly in the form.)

**Do you want to change the color of the combo box or list box?**

The *BackColor* property determines what the color of the box is. The *ForeColor* property determines the color of the text in the box.

**Do you want to change the font in the combo box or list box?**

Use the *Font* property.

**Do you want to make the combo box or list box inactive or make it invisible?**

The Enabled property determines whether the combo box or list box is active or not. (If the combo box or list box is inactive, it will be grayed out and nothing will happen when the end user clicks it.) The Visible property determines whether the combo box or list box is visible to the end user.

**Do you want to display a tool tip when the cursor is on top of the combo box or list box?**

Use the ToolTipText property.

**Do 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 6-110.

**Do you want the end user to be able to tab to the combo box or list box?**

Use the Tabstop property.

**Do 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 6-12.

## 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; 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 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. (Optional, but recommended.) Give your edit box or multi-edit box a more meaningful name than *EditBoxn* or *MultiEditBoxn*.
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*. 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* in Chapter 2, *Developing Procedures*.)

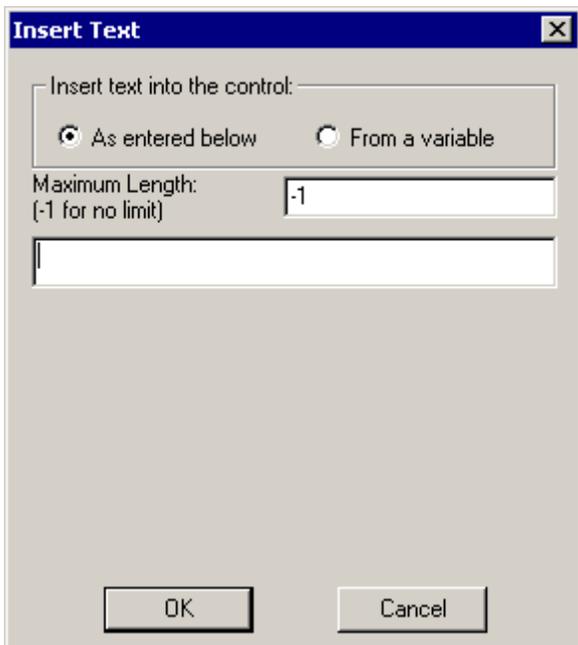
6. Click *OK*.

**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 variable's type. 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 *WebFOCUS Maintain Language Reference*.

**Reference** **Insert Text Dialog Box**

You use the Insert dialog box to populate edit and multi-edit boxes with data.



This dialog box contains the following elements:

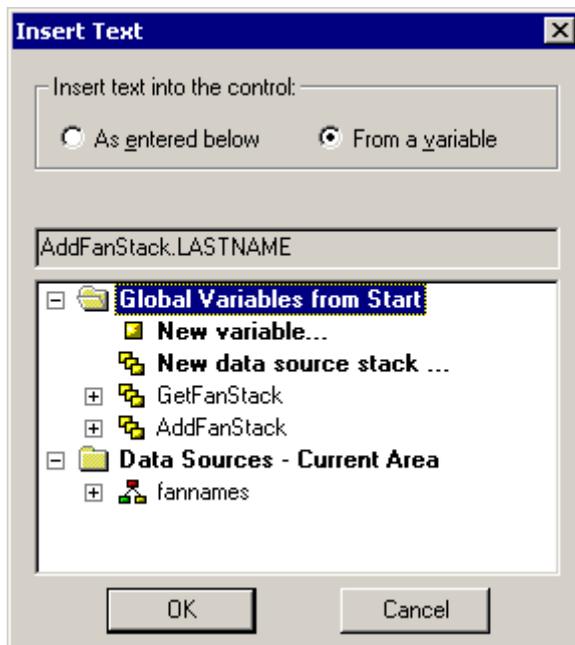
### Insert text into the control

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)

(Only if you selected *As entered below*.) Enter the maximum number of characters this edit box or multi-edit box can contain.



### New variable...

(Only if you selected *From a variable*.) Opens the New Variable dialog box, where you can create a variable.

### New data source stack...

(Only if you selected *From a variable*.) Opens the Stack Editor, where you can create a stack.

### List of data source stacks and variables in your procedure

(Only if you selected *From a variable*.) Contains a list of the existing stacks, variables, and database fields in your procedure.

Either select a variable, or expand a data source stack or database and select a column.

### **Data Sources - Current Area**

(Only if you selected *From a variable*.) Lists the fields from the data sources used in this procedure

**Note:** We recommend that you 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.

### **Do you want to change the contents of the edit box or multi-edit box?**

Use the Text property to open the Insert Text dialog box.

### **Do you want to change the name of the edit box or multi-edit box that identifies it to the procedure?**

Change the (Name) property.

### **Do you want to change the size or location of the edit box or multi-edit box?**

Change the Bottom, Left, Right, and Top properties. (You can also move or resize the edit box or multi-edit box directly in the form.)

### **Do you want to change the color of the edit box or multi-edit box?**

The BackColor property determines what the color of the box is. The ForeColor property determines the color of the text in the box.

### **Do you want to change the font in the edit box or multi-edit box?**

Use the Font property.

### **Do 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.

### **Do 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.)

### **Do you want to make the contents of the edit box or multi-edit box read-only?**

Use the ReadOnly property. (You could accomplish this by disabling the Enable property, but this method does not gray the control.)

### **Do you want to validate end user entry into an edit box?**

Use the Validation property. (This property does not apply to multi-edit boxes.)

**Do you want to remove the border or add a border to the edit box or multi-edit box?**

The Border property determines whether you have a border.

**Do you want to make the edit box or multi-edit box inactive or make it invisible?**

The Enabled property determines 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 grayed out and nothing will happen when the end user clicks it.) The Visible property determines whether the edit box or multi-edit box is visible to the end user.

**Do 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.

**Do you want to display a tool tip when the cursor is on top of the edit box or multi-edit box?**

Use the ToolTipText property.

**Do 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 6-110.

**Do you want the end user to be able to tab to the edit box or multi-edit box?**

Use the Tabstop property.

**Do 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 6-12.

**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.



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 you want to create prompted edit boxes for 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.

**Tip:** We recommend that you 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.

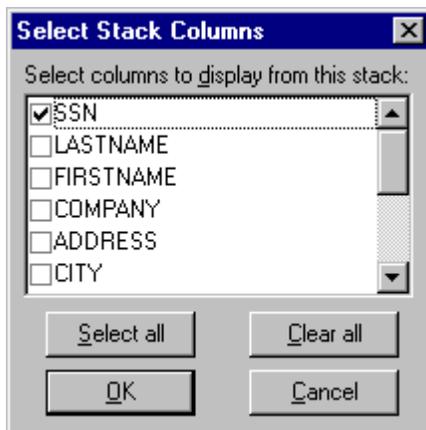
If you want to change the properties for either the edit box or the text control separately, select the prompted edit box. Then select either the edit box or the text control in the drop-down list of grouped controls in the property sheet.

For more information on grouping controls, see *Grouping Controls* on page 6-10.

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 6-9.

## **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. You use this dialog box to determine which columns in the data source stack to create prompted edit boxes for.



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**

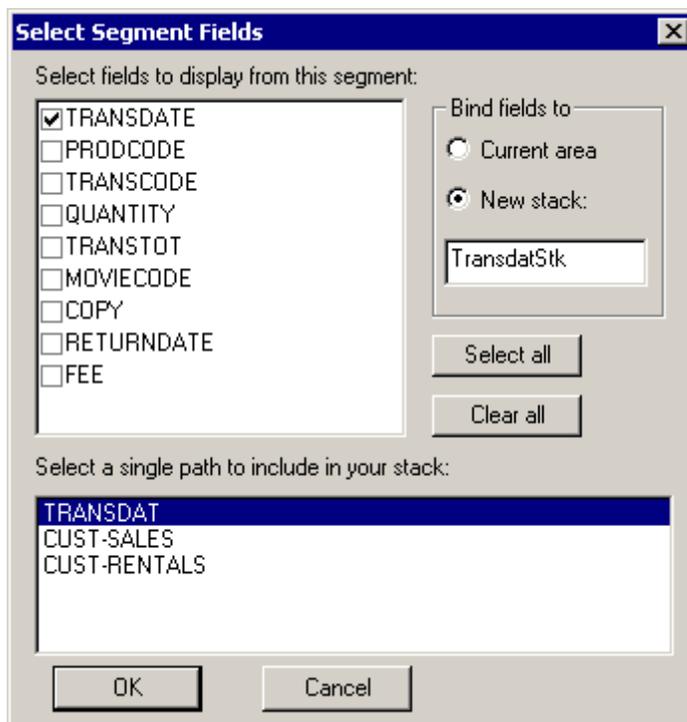
Clears all the columns in the data source stack.

**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. You 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, 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 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.

**Tip:** 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 to be, at approximately the size you want.
3. (Optional, but recommended.) Give your frame a more meaningful name than *Frame1*.
4. If necessary, readjust the size and placement of your frame.

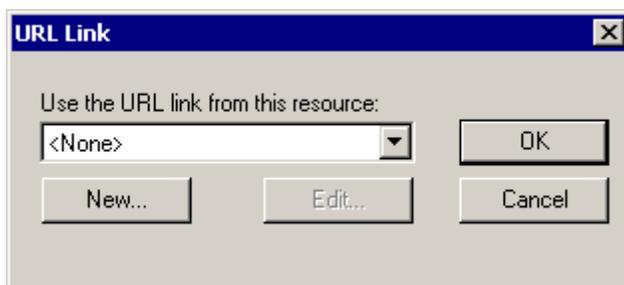
5. Double-click the frame, or select the Source property to open the URL Link dialog box.
6. If you have an existing Web link that you would like to use, select its name from the *Use the URL link from this resource* list.

Otherwise, click *New...* to create a new Web link. For more information, see *How to Create a Web Link* in Chapter 5, *Defining Events and Event Handlers*.

7. Click *OK*.

## Reference **URL Link Dialog Box**

You use this dialog box to specify the URL link for a frame on your form.



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.

### **Do you want to change the contents of the frame?**

Use the Source property.

### **Do you want to change the name of the frame that identifies it to the procedure?**

Change the (Name) property.

**Do you want to change the size or location of the frame?**

Change the Bottom, Left, Right, and Top properties. (You can also move or resize the frame directly in the form.)

**Do you want to add scroll bars to the frame?**

Set the Scrolling property to Auto or Yes.

**Do you want to add a border to the frame?**

The Border property determines whether you have a border.

**Do you want to make the frame invisible?**

The Visible property determines whether the frame is visible to the end user.

**Do you want to display a tool tip when the cursor is on top of the frame?**

Use the ToolTipText property.

**Do you want to move the frame to another layer?**

Use the Layer property. For more information, see *Layering Controls* on page 6-12.

## Using Grids

---

The Grid control places a read/write grid (stack editor) on a form that enables you to display and edit selected stack columns. The read/write grid differs from an HTML Table in that the grid allows you to directly change data in a stack.

The read/write grid is an ActiveX control. Installation of WebFOCUS Desktop 5.2 will automatically register the control on your machine to enable it for local deployment.

For Web deployment, the first time an application containing the Grid control is run it will require a one-time download of the signed ActiveX control for registration on the user's machine (the exception is if WebFOCUS Developer Studio is also installed on the user's machine). The browser security settings under Internet Options should allow a prompt or enable for downloading signed ActiveX controls. For more information, see the ReadMe file.

### **Procedure** How to Place a Read/Write Grid on a Form

1. Select the *Grid*  control in the Controls palette.
2. Draw a rectangle on your form approximately where you want to place your read/write grid, 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* in Chapter 2, *Developing Procedures*.)

4. Select the columns in the data source stack that you want to display and click . WebFOCUS Maintain adds these columns to the Column Selection box.  
To remove a column from the Column Selection box, select it and click .
5. If you want to remove row numbers, deselect *Include a row number column*.
6. If you want to change the order of the columns in the Column Selection box, select them and use  and .
7. To sort your columns by title, column, or width, click the corresponding button (*Header Title, Column, or Width*) in the Column Selection box.
8. To format a column, double-click it in the Column Selection box to open the Grid Column Properties dialog box.
9. Click *OK* to leave the Grid Control Columns dialog box.
10. Optional, but recommended.) Give your read/write grid a more meaningful name than *gridn*.
11. If necessary, readjust the size and placement of your grid.

### **Example** Using a Read/Write Grid in a Form

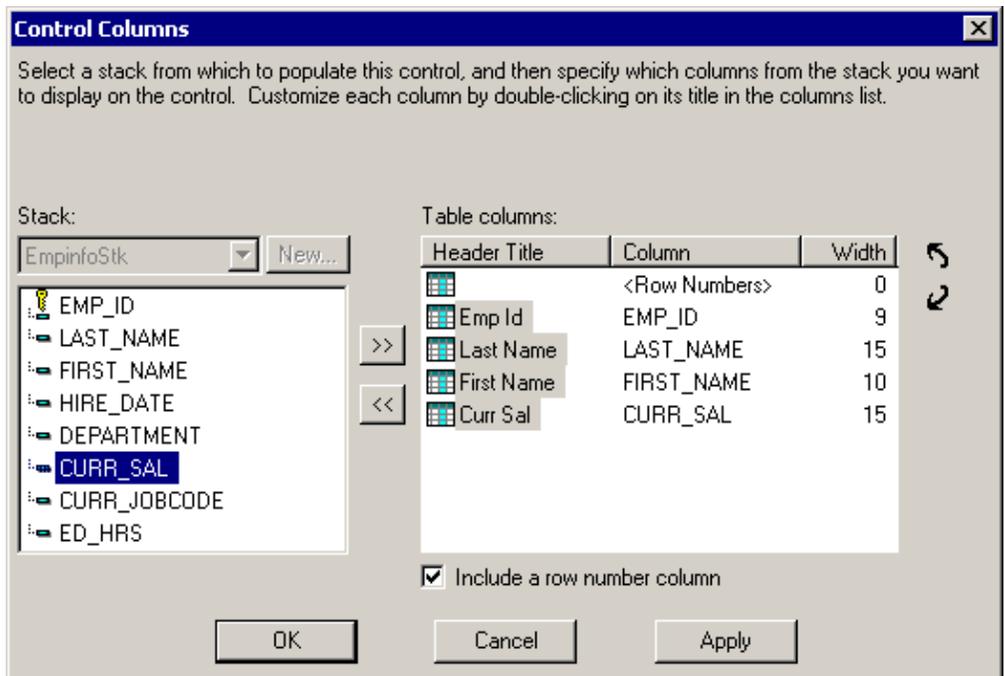
Suppose that you are creating a read/write grid to display information in an application built on the EMPLOYEE data source.

1. Create a Maintain procedure.
2. Use the EMPLOYEE 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 employee and click the  button.

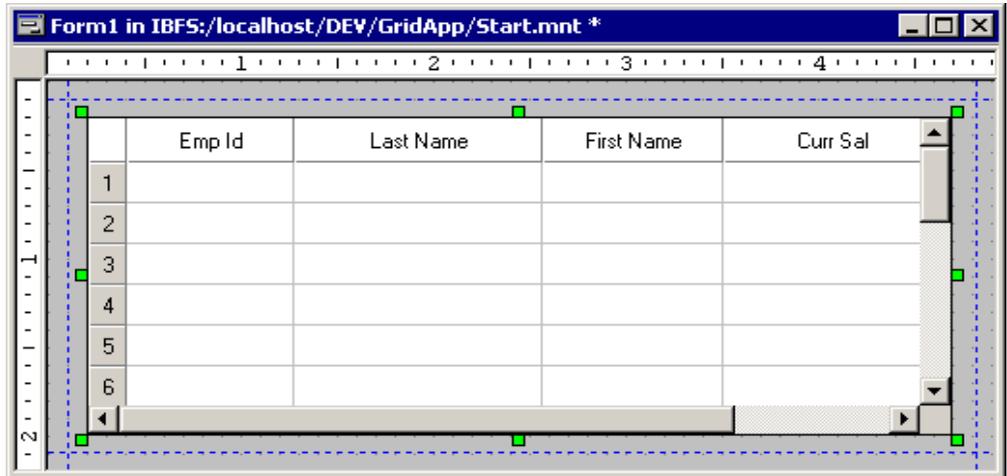
If employee does not appear in the project path, then see *How to Add an Existing File to Your Project* in Chapter 1, *Working in the Maintain Development Environment*.

3. Create a new data source stack by right-clicking the Start procedure, clicking *New*, and then clicking *Data source stack...*
4. In the Stack Editor, name the stack EmpinfoStk.
5. In the Available fields box, expand the EMPLOYEE data source, and then the EMPINFO segment.

6. Move the EMP\_ID field to the *Stack columns* box.
7. Click *OK*. You have now created the stack that will populate your read/write grid.
8. Open Form1.
9. Select the *Grid*  control on the Controls palette.
10. Draw a reasonably large rectangle on Form1.
11. WebFOCUS Maintain automatically opens the Control Columns dialog box. Make sure EmpinfoStk is selected in the list of data source stacks.
12. Copy EMP\_ID, LAST\_NAME, FIRST\_NAME, and CURR\_SAL to the Column selection box.



13. Click *OK*. The grid will appear on your form with the columns you selected.



When you run your application, the read/write grid on Form1 will display the contents of EmpinfoStk. The user can update the EMPLOYEE data source by modifying information in the grid and updating.

### **Reference** Events Available in a Read/Write Grid

The following table lists the events available for the read/write grid.

**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 or VB Script.

#### **Blur**

Occurs when the cursor or focus is moved to another cell or object. This event was 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 was 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 was 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-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 a cell changes focus

**OnCharDown**

Occurs when the end user hits 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 will need to assign each column a specific width. Maintain triggers and IWC triggers are not supported for this event.

**OnEditFinish**

This notification allows 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 a 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 hits 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 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. Once 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 Read/Write Grid

When you select your read/write grid, you will see a list of read/write grid properties in the property sheet. Changing these properties will change what your read/write grid looks like and how it behaves at run time.

### **Do you want to change the name that identifies the read/write grid to the procedure?**

Change the (Name) property.

### **Do you want to change which columns appear in the read/write grid?**

Use the Columns property to open the *Grid Control Columns* dialog box.

### **Do you want to change the size or location of the read/write grid?**

Change the Bottom, Left, Right, and Top properties. (You can also move or resize the grid directly in the form.)

### **Do you want to omit the header?**

Use the Headers property.

### **Do you want to change the color of the read/write grid?**

The BackColor property determines what color the background is. The ForeColor property determines the color of the text. The AlternateRowColor property turns every other row a different color. The HeaderBackColor determines what color the background of the header is. The HeaderForeColor property determines what color the text of the header is. You can also use the Columns property to open the *Grid Control Columns* dialog box, and then double-click a column to open the Grid Column Properties dialog box, where you can set colors for individual columns.

### **Do you want to change the text font?**

Use the Font and the HeaderFont property. You can also use the Columns property to open the *Grid Control Columns* dialog box, and then double-click a column to open the Grid Column Properties dialog box, where you can set fonts for individual columns.

### **Do you want to control which columns remain stationary while scrolling the table?**

Use the Fixed Columns property.

### **Do you want to display lines between columns and rows?**

Use the Grid Lines property.

### **Do you want to make the read/write grid inactive or make it invisible?**

The Enabled property determines whether the *read/write grid* is active or not. (If the *read/write grid* is inactive, it will be grayed out and nothing will happen when the user clicks it.) The Visible property determines whether the *read/write grid* is visible to the user.

**Do you want to change what the cursor looks like when it is on top of the read/write grid?**

Use the CursorPointer property.

**Do you want to display a tool tip when the cursor is on top of the read/write grid?**

Use the ToolTipText property.

**Do you want to assign a help topic to the read/write grid?**

Use the Help property. For more information, see *Assigning Help to Your Forms and Controls* on page 6-110.

**Do you want to move the read/write grid to another layer?**

Use the Layer property. For more information, see *Layering Controls* on page 6-12.

**Using Functions in the Read/Write Grid**

Within the read/write grid, you can use VBScript functions to alter the properties of the grid. Examples are given for the most commonly used functions. For a complete list of functions available in the read/write grid, see *Property Functions in the Read/Write Grid* on page 6-47.

**Reference Property Functions in the Read/Write Grid**

Name of Function	VBScript	Result
AddFont	<code>fontid=Formn.Gridn.AddFont (fontid, 0, 1)</code>	Text in the cell indicated by column and row number will appear in the font indicated by fontid
QuickSetFont	<code>Formn.Gridn.QuickSetFont c, r, fontid</code>	Sets font for specified cells
RedrawAll	<code>Formn.Gridn.RedrawAll</code>	Updates grid when you have made changes.
AddTab	<code>Formn.Gridn.AddTab "text", n</code>	When the tab is clicked the sheet will display the text in cell n.
CellGetReadOnly CellSetReadOnly	<code>Formn.Gridn.GetCell c, r, Formn.Gridn.CellSetReadOnly [True False] read=Formn.Gridn.CellGetReadOnly</code>	Sets specified cell to read only access when set to True.
CellGetText CellSetText	<code>Formn.Gridn.GetCell c, r, Formn.Gridn.GetCellText Formn.Gridn.CellSetText "text"</code>	Retrieves the text of a given cell and sets the text of that cell.

Name of Function	VBScript	Result
QuickSetText	<code>Formn.Gridn.QuickSetText c, r, "text"</code>	Serves as an alternative method to using CellGetText and CellSetText individually.
CellGetTextColor	<code>color = Formn.Gridn.CellGetTextColor(r g b)</code>	Specifies the color of the text in a cell. Values for r, g, and b specify the desired intensity of red, green, and blue. The values are on a scale of 0 to 255.
QuickSetTextColor	<code>Formn.Gridn.QuickSetTextColor 0, 0, RGB(r,g,b)</code>	Sets the text color of the specified cell
QuickSetBackColor	<code>Formn.Gridn.QuickSetBackColor 0, 0, RGB(r,g,b)</code>	Sets the background color of the specified cell
GetCurrentColumn	<code>column=Formn.Gridn.GetCurrentColumn</code>	Identifies the column that is clicked on or selected by user.
GetCurrentRow	<code>row=Formn.Gridn.GetCurrentRow</code>	Identifies the row that is clicked on or selected by user.
GetGridDefault	<code>Formn.Gridn.GetGridDefault</code>	Default grid is set to 10 columns and 10 rows.
QuickSetMask	<code>Formn.Gridn.QuickSetMask (0, 0, "###");</code>	Selectively includes or excludes certain values from a field. For information on how to use Mask options, see <i>How to Use The Mask Function</i> on page 6-49.
QuickSetAlignment	<code>Formn.Gridn.QuickSetAlignment 0, 0, 2</code>	Sets the alignment of the specified cells
GotoCell	<code>Formn.Gridn.GotoCell c, r,</code>	The cell specified by column and row number will be in focus
GotoColumn	<code>Formn.Gridn.GotoColumn c</code>	The column specified by column and row number will be inn focus
LockColumns	<code>Formn.Gridn.LockColumns c</code>	The specified number of columns remain visible when scrolling horizontally.

Name of Function	VBScript	Result
LockRows	<code>Formn.Gridn.LockRows r</code>	The specified number of rows remain visible when scrolling vertically.
SetCurrentColumn	<code>Formn.Gridn.SetCurrentColumn c</code>	Focus will be on the column indicated by c.
SetCurrentRow	<code>Formn.Gridn.SetCurrentRow r</code>	Focus will be on the row indicated by r.
SetGridDefault	<code>Formn.Gridn.SetGridDefault</code>	Allows you to specify the default values for a grid.
SetGridDefault	<code>Formn.Gridn.SetGridDefault</code>	Allows you to specify the default values for a grid.

**Syntax**      **How to Use The Mask Function**

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:

```
Formn.Gridn.QuickSetMask (0, 0, "#");
```

Where # can equal any combination of the following values:

- 0. Digit (0 through 9), entry required; plus and minus signs not allowed.
- 9 Digit or space (entry optional; plus and minus signs not allowed)
- L Letter (A through Z, entry required)
- ? Letter (A through Z, entry optional)
- A Letter or Digit (Entry required)
- a Letter or Digit (Entry optional)
- & Any character or a space (Entry required)
- C Any character or a space (Entry optional)
- . , : ; - / Decimal placeholder and thousands, date, and time separators. (The actual character used will depend on the regional settings specified in the Windows Control Panel)

`\\` 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, `\\&` is displayed as just &, etc.)

For an example of how the Mask function can be used, see *Setting a Mask in a Grid Cell* on page 6-52.

### **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:

```
fontid = Form1.Grid1.AddFont (Garamond, 14, 1)
Form1.Grid1.QuickSetFont 0,0,fontid
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:

```
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:

```
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, etc.

**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:

```
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, etc.

**Example**    **Setting the GoToCell Function**

To set the GoToCell function, code the following VBScript within the push button event handler:

```
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:

```
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 CurrentColumn function, code the following VBScript within the push button event handler:

```
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:

```
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:

```
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.CellSetText "NewText"  
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.

```
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.

**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 approximately where you want your group box to be 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 6-10.

## 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.

### **Do you want to change the label at the top of the group box?**

Change the Text property.

### **Do you want to change the name of the group box that identifies it to the procedure?**

Change the (Name) property.

### **Do you want to change the size or location of the group box?**

Change the Bottom, Left, Right, and Top properties. (You can also move or resize the group box directly in the form.)

### **Do you want to change the color of the group box?**

The BorderColor property determines what the color of the border is. The ForeColor property determines the color of the text in the label.

### **Do you want to change the label font?**

Use the Font property.

### **Do you want to change the alignment of the label (left-justified, centered, or right-justified)?**

Use the Alignment property.

### **Do you want to change the border type or width?**

The Border property determines the type of border (none, normal, 3D, and so on). The BorderWidth property determines the width of the border.

### **Do you want to make the group box inactive or make it invisible?**

The Enabled property determines whether the group box is active or not. (If the group box is inactive, it will be grayed out and nothing will happen when the end user clicks it.) The Visible property determines whether the group box is visible to the end user.

### **Do you want to change what the cursor looks like when it is on top of the group box?**

Use the CursorPointer property.

### **Do you want to display a tool tip when the cursor is on top of the group box?**

Use the ToolTipText property. The tool tip applies to the whole inner area, including any controls within it (unless these controls have their own tool tips).

**Do 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 6-110.

**Do you want to move the group box to another layer?**

Use the Layer property. For more information, see *Layering Controls* on page 6-12.

## 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 an EDA table services report and integrate its output into a form.

The position of the HTML object's top-left corner on the form determines the starting position of the HTML code once it is displayed 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 *WebFOCUS Maintain Advanced Tutorial* in *WebFOCUS Maintain Getting Started*.

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 Chapter 5, *Defining Events and Event Handlers*.

### **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 approximately where you want your HTML Object to be at approximately the size you want.
3. (Optional, but recommended.) Give your HTML Object a more meaningful name than HTMLn.
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.

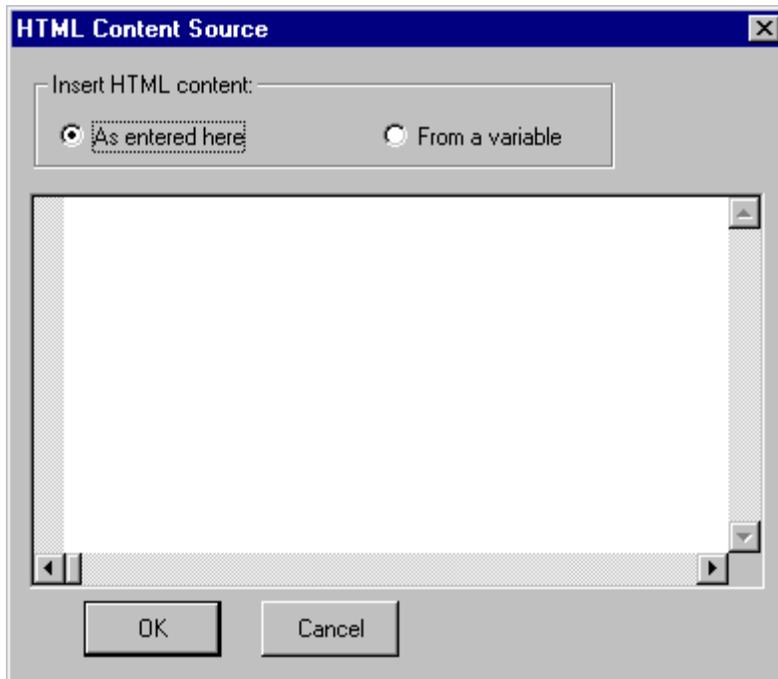
To assign the content of the HTML Object to a variable, select *From a variable*. Then select a variable or data source stack column. (You can create a new variable or data source stack by clicking *New variable...* or *New data source stack...* For more information, see *How to Create a Variable in a Procedure* in Chapter 2, *Developing Procedures* or *How to Create a Data Source Stack Explicitly Using the Stack Editor* in Chapter 2, *Developing Procedures*.

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 to define the contents of the HTML Object.

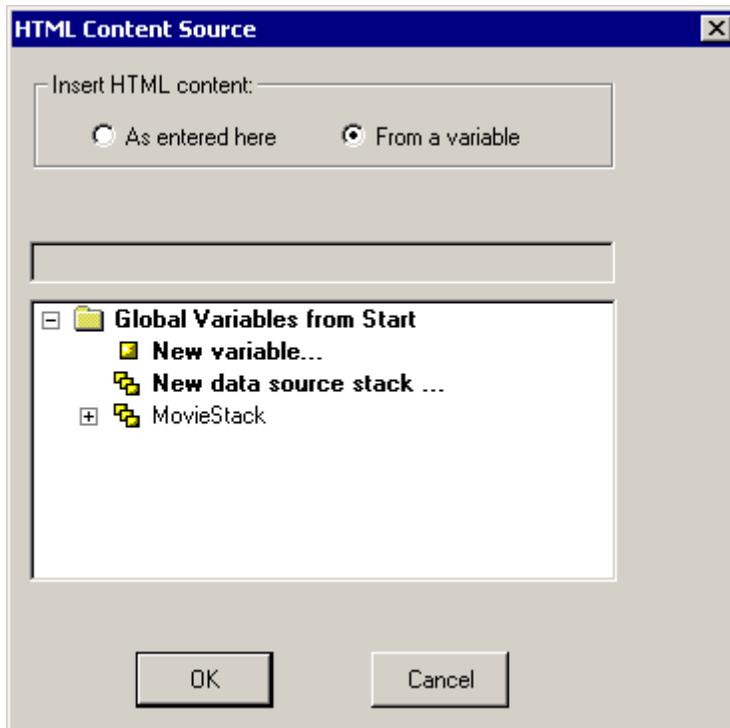


This dialog box contains the following elements:

### Insert HTML content

Select *As entered here* to set the contents directly.

Select *From a variable* to set the contents from a variable.



### New variable...

(Only if you selected *From a variable*.) Opens the New Variable dialog box, where you can create a variable with the data type *Stack of...*

### New data source stack...

(Only if you selected *From a variable*.) Opens the Stack Editor, where you can create a stack.

### List of data source stacks and variables in your procedure

(Only if you selected *From a variable*.) 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.

### **Do you want to change the contents of the HTML Object?**

Change the Content property or change the contents of the stack column or variable it is bound to.

### **Do you want to change the name of the HTML Object that identifies it to the procedure?**

Change the (Name) property.

### **Do you want to change the size or location of the HTML Object?**

Change the Bottom, Left, Right, and Top properties. (You can also move or resize the HTML Object directly in the form.) Note that if the Overflow property is set to Visible, the HTML Object will be as wide and as long as it needs to be to display everything.

### **Do you want to restrict the size of the HTML Object?**

Set the Overflow property to Clip or Scroll.

### **Do you want to change the color of the HTML Object?**

The BackColor property determines what color the background is. The ForeColor property determines the color of the text.

### **Do you want to change the text font?**

Use the Font property.

### **Do you want to add a border to the HTML Object?**

The Border property determines whether you have a border. The BorderColor property determines the color of the border. The BorderWidth property determines the width of the border.

### **Do you want to make the HTML Object inactive or make it invisible?**

The Enabled property determines whether the HTML Object is active or not. (If the HTML Object is inactive, it will be grayed out and nothing will happen when the end user clicks it.) The Visible property determines whether the HTML Object is visible to the end user.

### **Do you want to change what the cursor looks like when it is on top of the HTML Object?**

Use the CursorPointer property.

### **Do you want to display a tool tip when the cursor is on top of the HTML Object?**

Use the ToolTipText property.

**Do 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 6-110.

**Do you want to move the HTML Object to another layer?**

Use the Layer property. For more information, see *Layering Controls* on page 6-12.

## Using HTML Tables

---

An HTML Table displays the contents of a data source stack to end users. End users can view data and click on columns to fire events.

**Tip:** Avoid placing controls beneath the table because the number of rows it contains at run time determines the height of the table. If the number of rows is more than you expected, the controls that you thought were beneath the table may be on top of the table.

To avoid this problem, you can also set the HTML table's Overflow property to Clip or Scroll to restrict the size of the table.

### *Procedure* How to Place an HTML Table on Your Form

1. Select the *HTML Table*  control in the Controls palette.
2. Draw a rectangle on your form approximately where you want your HTML Table to be at approximately the size you want.
3. WebFOCUS Maintain automatically opens the HTML Table Columns dialog box. (You can also open this dialog box by double-clicking the HTML Table 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* in Chapter 2, *Developing Procedures*.)
4. Select the columns in the data source stack that you want to display and click . WebFOCUS Maintain adds these columns to the Table Columns box.

To remove a column from the Table Columns box, select it and click .

5. If you want to remove row numbers, deselect *Include a row number column*.
6. If you want to change the order of the columns in the Table Columns box, select them and use  and .
7. To sort your columns by title, column, or width, click the corresponding button (Header Title, Column, or Width) in the Table Columns window.

8. To format a column, double-click it in the Table Columns window to open the Table Column dialog box.
9. Click *OK* to leave the HTML Table Columns dialog box.
10. (Optional, but recommended.) Give your HTML Table a more meaningful name than HTMLTablen.
11. If necessary, readjust the size and placement of your 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.

### **Example** Using an HTML Table to Display the Results of a WebFOCUS Report

Suppose you have created the following report using WebFOCUS Developer Studio and included it in your project:

```

-* 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 <TABPAGE NO  "
ON TABLE NOTOTAL

-* This command directs WebFOCUS to hold the report results in a stack.
ON TABLE PCHOLD

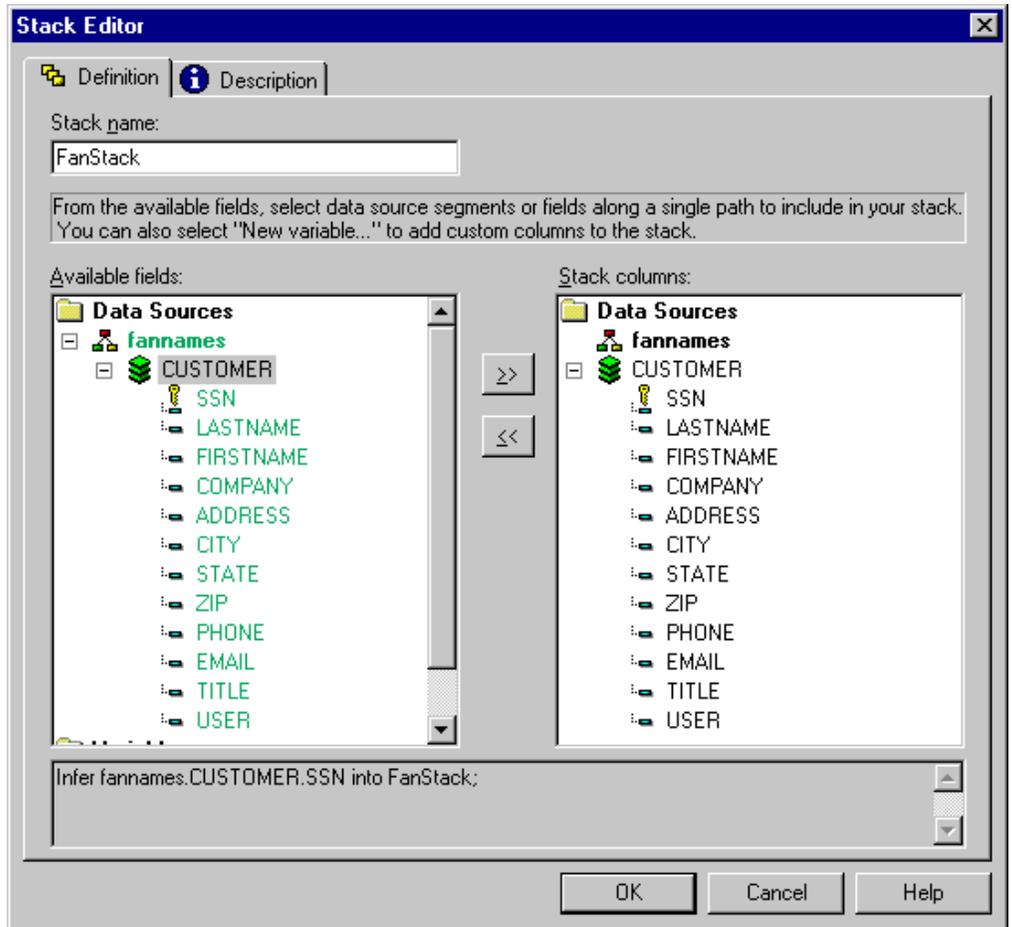
ON TABLE SET STYLE *
UNITS=IN, PAGESIZE='Letter', LEFTMARGIN=0.250000, RIGHTMARGIN=0.250000,
TOPMARGIN=0.250000, BOTTOMMARGIN=0.250000, SQUEEZE=ON,
ORIENTATION=PORTRAIT, $
TYPE=REPORT, GRID=ON, FONT=TIMES NEW ROMAN, SIZE=10, COLOR=BLACK,
BACKCOLOR=NONE, STYLE=NORMAL, $
ENDSTYLE
END
```

This report displays the Firstname, Company, and Email fields from the Fannames data source, sorted by the Lastname field. Notice the ON TABLE PCHOLD syntax.

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  button.  
If fannames does not appear in the project path, then see *How to Add an Existing File to Your Project* in Chapter 1, *Working in the Maintain Development Environment*.
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. Move the SSN field to the *Stack columns* box.



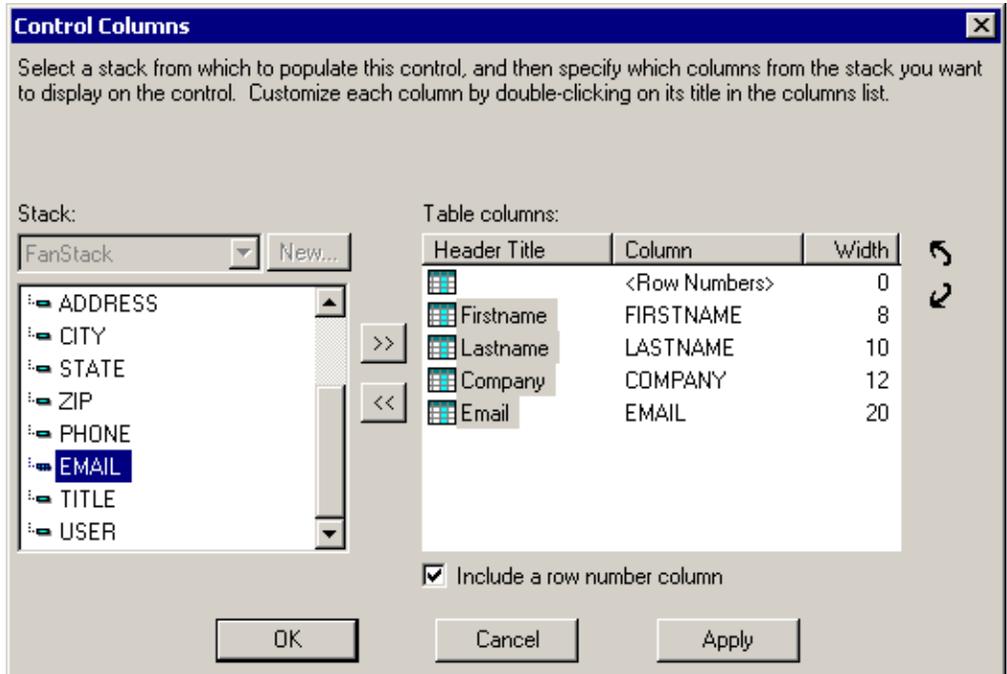
7. Click **OK**. You have now defined the data source stack that will receive the contents of the WebFOCUS report.
8. Open the Maintain procedure in the Procedure Editor.
9. After the line of Maintain code that defines FanStack (INFER...), but before the Maintain code that displays Form1 (WINFORM SHOW...), enter the following Maintain code:

```
EXEC FANREPT INTO FANSTACK;
```

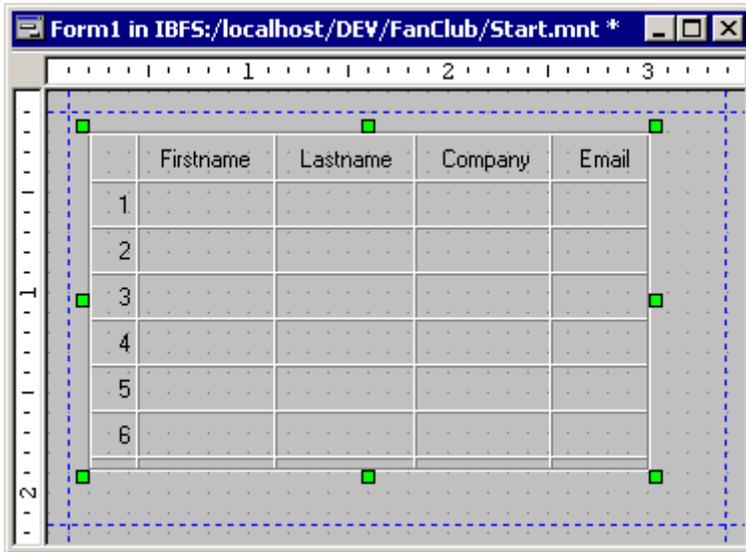
You can type all of this code directly, or you can generate the EXEC FANREPT part either with the Language Wizard or by dragging FANREPT from the Project Explorer into the Procedure Editor. You will have to type INTO FANSTACK by hand.

10. Open Form1.

11. Select the HTML Table  control on the Controls palette.
12. Draw a reasonably large rectangle on Form1.
13. WebFOCUS Maintain automatically opens the Control Columns dialog box. Make sure FanStack is selected in the list of data source stacks.
14. Copy FIRSTNAME, LASTNAME, COMPANY, and EMAIL to the Table columns box.



15. Click OK.



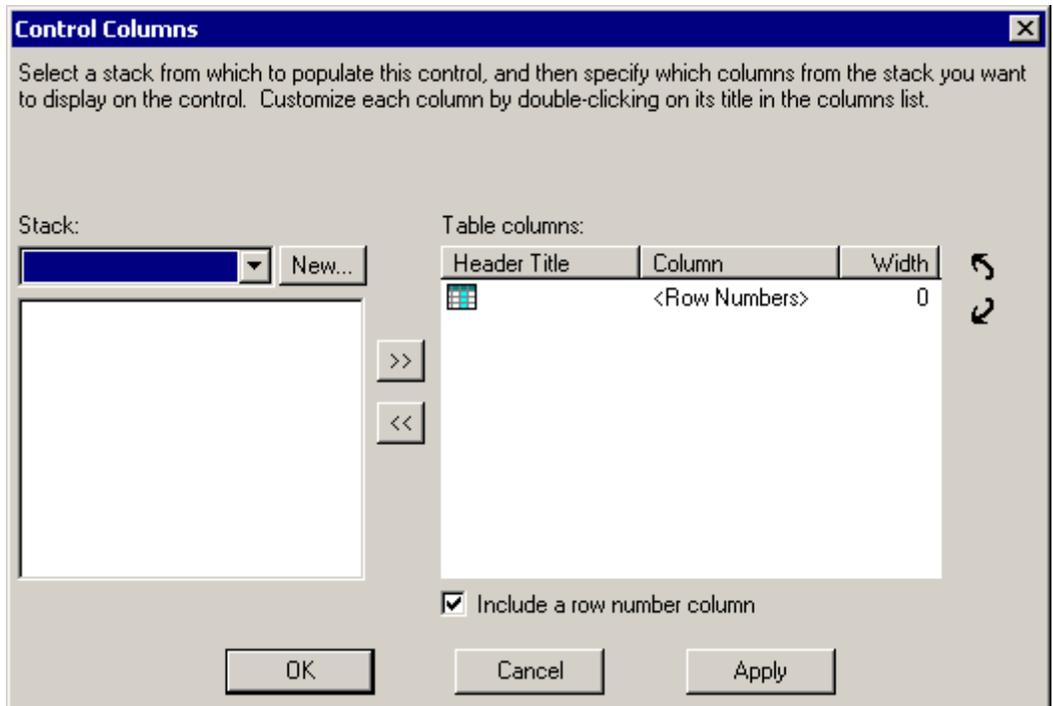
Now, when you run your application, WebFOCUS Maintain executes FANREPT and returns the results to FanStack. The HTML Table on Form1 displays the contents of FanStack.

Notice that any formatting information in the report is lost. (If you wish, you can apply new formatting using the Table Column dialog box.) If you want to view formatting inherent in the report, you would use the ON TABLE HOLD HTMTABLE option instead of ON TABLE PCHOLD and view the results in an HTML Object.

## Reference Control Columns Dialog Box

You use the Control Columns dialog box to populate an HTML Table with data.

You also use the Control Columns dialog box to populate a Grid with data. For more information, see *Using Grids* on page 6-40.



This dialog box contains the following elements:

### Stack

Displays the data source stacks available in this procedure.

### New...

Enables you to create 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.

**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.

**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**

You use the Table Column dialog box to format columns in your HTML Table.

You use the Grid Column Properties dialog box to format columns in your grid. For more information, see *Using Grids* on page 6-40.

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. To put a line break in the column title, type `\n` where you want the break.

### **Width in characters**

Contains the width of the column. You can type a new value here.

### **Justification**

you to left-justify, center, or right-justify the contents of a column.

### **Content Type**

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 need to use double backslashes (for example, C:\ibi\_img\graphic.gif).

### **Links**

(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 6-68.

### **Read Only**

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**

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 6-108.

### **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 6-108.

## **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 Web pages.

### ***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 the end user 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 6-69.

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 Chapter 5, *Defining Events and Event Handlers*.

**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 for an HTML Table's Data Source Stack* on page 6-70.

## 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 for an HTML Table's Data Source Stack

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).

## **Changing HTML Table Properties**

When you select your HTML Table, you will see a list of HTML Table properties in the property sheet. Changing these properties will change what your HTML Table looks like and how it behaves at run time.

### **Do you want to change the contents of the HTML Table?**

Use the Columns property to open the HTML Table Columns dialog box.

### **Do you want to change the name that identifies the HTML Table to the procedure?**

Change the (Name) property.

### **Do you want to change the size or location of the HTML Table?**

Change the Bottom, Left, Right, and Top properties. (You can also move or resize the HTML Table directly in the form.) Note that if the Overflow property is set to Visible, the HTML Table will be as wide and as long as it needs to be to display everything.

### **Do you want to restrict the size of the HTML Table?**

Use the Overflow property.

### **Do you want to omit the header?**

Use the Headers property.

### **Do you want to change the color of the HTML Table?**

The BackColor property determines what color the background is. The ForeColor property determines the color of the text. The AlternateRowColor property turns every other row a different color. The HeaderBackColor determines what color the background of the header is. The HeaderForeColor property determines what color the text of the header is. You can also use the Columns property to open the Control Columns dialog box, and then double-click a column to open the Table Column dialog box, where you can set colors for individual columns.

**Do you want to change the text font?**

Use the Font and the HeaderFont property. You can also use the Columns property to open the Control Columns dialog box, and then double-click a column to open the Table Column dialog box, where you can set fonts for individual columns.

**Do you want to add a border or grid lines to the HTML Table?**

The Border property determines whether you have a border and what type of border it is. The BorderColor property determines the color of the border. The BorderWidth property determines 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. The GridLines property determines whether the HTML Table has gridlines.

**Do you want to make the HTML Table inactive or make it invisible?**

The Enabled property determines whether the HTML Table is active or not. (If the HTML Table is inactive, it will be grayed out and nothing will happen when the end user clicks it.) The Visible property determines whether the HTML Table is visible to the end user.

**Do you want to change what the cursor looks like when it is on top of the HTML Table?**

Use the CursorPointer property.

**Do you want to display a tool tip when the cursor is on top of the HTML Table?**

Use the ToolTipText property.

**Do you want to assign a help topic to the HTML Table?**

Use the Help property. For more information, see *Assigning Help to Your Forms and Controls* on page 6-110.

**Do you want to move the HTML Table to another layer?**

Use the Layer property. For more information, see *Layering Controls* on page 6-12.

## 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's 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 6-73 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, you need to 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* and *How to Create a Data Source Stack Explicitly Using the Stack Editor* in Chapter 2, *Developing Procedures*.

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.) Give your image a more meaningful name than *Imagen*.
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*.

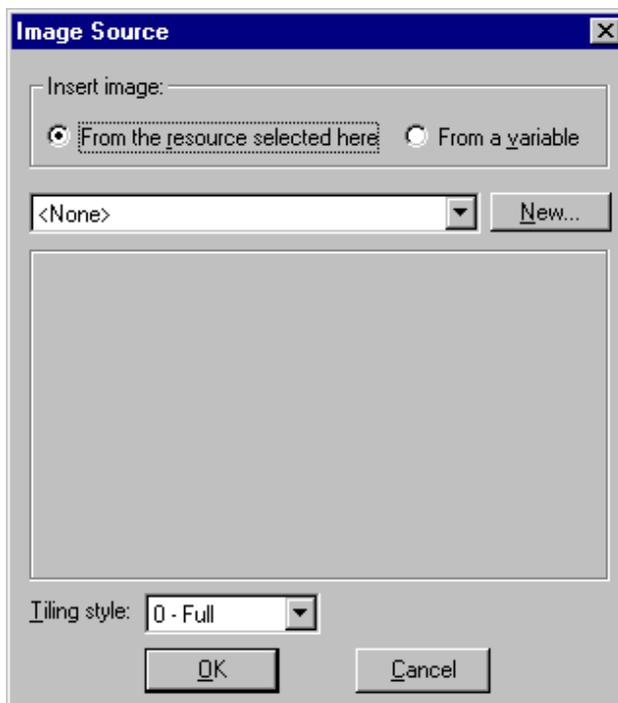
If you prefer to browse for the image, click the  button. The Resource Wizard will automatically preview the image you select.

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

You use the Image Source dialog box 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. To add images to the list, click *New...*

### **New...**

Opens the Resource Wizard so that you can add new images to your project.

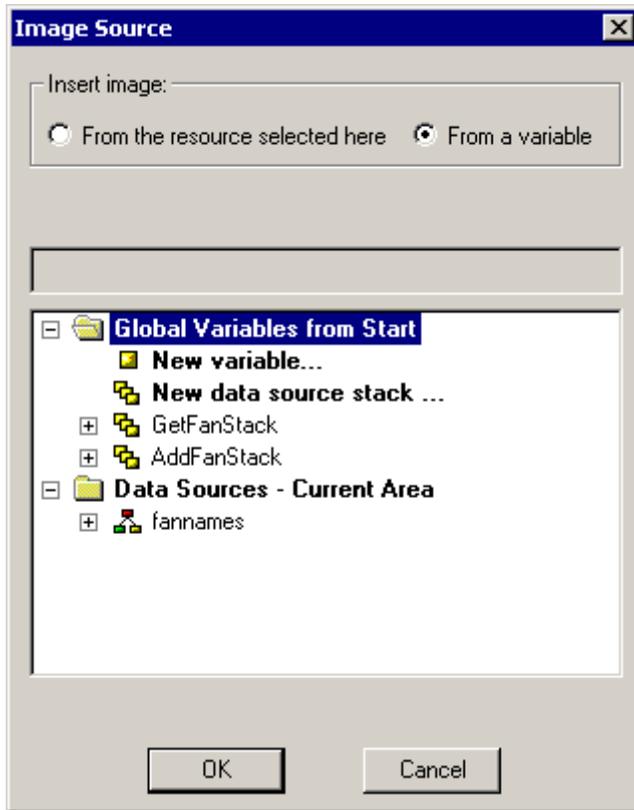
### **Tiling style**

(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 top left corner of the form window.



**New variable...**

(Only if you selected *From a variable.*) Opens the New Variable dialog box, where you can create a variable.

**New data source stack...**

(Only if you selected *From a variable.*) Opens the Stack Editor, where you can create a stack.

**List of data source stacks and variables in your procedure**

(Only if you selected *From a variable.*) 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**

(Only if you selected *From a variable*.) Lists the fields from the data sources used in this procedure.

**Note:** We recommend that you 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 web sites out there that provide free images for buttons and backgrounds.
- 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 your end users' modems will be.

Browsers download HTML/DHTML first and then images. You can see 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 6-76.

- Use smaller image sizes and the appropriate image type—.jpps 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.

### **Do you want to change the name of the image that identifies it to the procedure?**

Change the (Name) property.

### **Do you want to change which image will appear?**

Use the Image property to open the Image Source dialog box.

### **Do you want to change the size or location of the image?**

Set the Stretched property to 1 - Yes and change the Bottom, Left, Right, and Top properties. (You can also move or resize the image directly in the form.)

**Do you want to add a border to the image?**

The Border property turns image borders on and off. The BorderColor property determines the border's color. The BorderWidth property determines the border's width.

**Do you want to change what the image looks like when the end user clicks it?**

Use the ImageDown property. For more information, see *Changing Images at Run Time Using ImageDown and ImageOver* on page 6-80.

**Do you want to make different regions of the image clickable?**

Use the Map property. For more information, see *Using Image Maps* on page 6-77.

**Do you want to make the image inactive or make it invisible?**

The Enabled property determines whether the image is active or not. (If the image is inactive, nothing will happen when the end user clicks it.) The Visible property determines whether the image is visible to the end user.

**Do you want to change what the cursor looks like when it is on top of the image?**

Use the CursorPointer property.

**Do 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 6-80.

**Do you want to display a tool tip when the cursor is on top of the image?**

Use the ToolTipText property.

**Do 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 6-110.

**Do you want to move the image to another layer?**

Use the Layer property. For more information, see *Layering Controls* on page 6-12.

## 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.

Once 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.



If you need to delete an area, use the Delete and Delete all buttons.



4. If you wish, rename your areas and apply tool tips 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**

You use the Image Map dialog box to define clickable locations in an image.



This dialog box contains the following elements:

### **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**

Is where you create and edit 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 in *How to Add an Image to Your Form* on page 6-72.

**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 has dependencies, you need to make sure that CLASSPATH on the end users' browsers are set to locate them.

**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.

If the Java applet you want is not part of your project, click *New...* to open the Resource Wizard. See *How to Add a Java Applet to Your Project as a Resource* on page 6-82 for more information.

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. (Optional, but recommended.) Give your Java applet a more meaningful name than `JavaAppletn`.

### **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  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:

```
Form.JavaApplet.params(n).Name="name"
```

To change the value for a Java applet parameter, use the following syntax

```
Form.JavaApplet.params(n).Value="value"
```

where:

*Form*

Is the name of the form the Java applet control is residing on.

*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

You use the Java Applet Parameters dialog box to define the behavior of a Java applet control that you have placed on your form.



This dialog box contains the following elements:

**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.



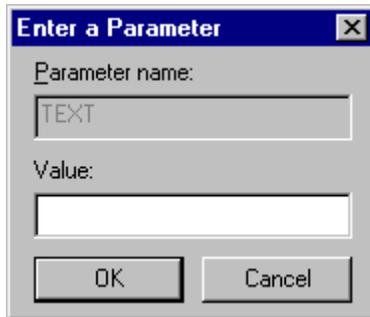
Moves a selected parameter down in the list of parameters.

**Reference Enter a Parameter Dialog Box**

You use the Enter a Parameter dialog box to define parameters for Java Applets, and to assign values to parameters when you place Java applets on your forms.

You 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 Chapter 5, *Defining Events and Event Handlers*.

You also use the Enter a Parameter dialog box to define parameters and values for a CGI or ISAPI program that you are defining as a Web link. For more information, see *How to Use the URL Wizard to Define an HTTP Web Link* in Chapter 5, *Defining Events and Event Handlers*.



This dialog box has the following components:

#### **Parameter name**

If you are defining parameters for a Java applet, WebFOCUS report, or CGI or ISAPI program, 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.

### **Do you want to change the name of the Java applet that identifies it to the procedure?**

Change the (Name) property.

### **Do you want to change which Java applet you place here?**

Use the Applet property to open the Java Applet Parameters dialog box.

### **Do you want to change the values for the Java applet parameters?**

Use the Applet property to open the Java Applet Parameters dialog box.

**Do you want to change the size or location of the Java applet?**

Change the Bottom, Left, Right, and Top properties. (You can also move or resize the Java applet directly in the form.)

**Do you want to add a border to the Java applet?**

The Border property turns Java applet borders on and off. The BorderColor property determines the border's color. The BorderWidth property determines the border's width.

**Do you want to make the Java applet inactive or make it invisible?**

The Enabled property determines whether the Java applet is active or not. (If the Java applet is inactive, nothing will happen when the end user clicks it.) The Visible property determines whether the Java applet is visible to the end user.

**Do you want to change what the cursor looks like when it is on top of the Java applet?**

Use the CursorPointer property.

**Do you want to display a tool tip when the cursor is on top of the Java applet?**

Use the ToolTipText property.

**Do 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 6-110.

**Do you want to move the Java applet to another layer?**

Use the Layer property. For more information, see *Layering Controls* on page 6-12.

## 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.

### **Do you want to change the name of the line that identifies it to the procedure?**

Change the (Name) property.

### **Do you want to change the size or location of the line?**

Change the Bottom, Left, Right, and Top properties. (You can also move or resize the line directly in the form.)

### **you want to change the line's color or width?**

The ForeColor property determines the line's color. The PenWidth property determines the line's width.

### **you want to make the line inactive or make it invisible?**

The Enabled property determines whether the line is active or not. (If the line is inactive, it will be grayed out and nothing will happen when the end user clicks it.) The Visible property determines whether the line is visible to the end user.

### **you want to change what the cursor looks like when it is on top of the line?**

Use the CursorPointer property.

### **you want to display a tool tip when the cursor is on top of the line?**

Use the ToolTipText property.

### **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 6-110.

### **you want to move the line to another layer?**

Use the Layer property. For more information, see *Layering Controls* on page 6-12.

## 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 6-88.
- Creating the menu item(s) 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 6-88.

### **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. (Optional, but recommended.) Give your Menu Bar a more meaningful name than *MenuBar1*.
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 sub menus 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 6-88.
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 information on how to place a menu bar on your form, see *How to Place a Menu Bar on Your Form* on page 6-88.

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 *Add to list*  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 sub menu, select the name of the pull-down menu to which you are adding the sub menu. Repeat steps 1-3 for each sub menu desired.
6. Repeat steps 1-4 until all desired pull-down menus and sub menus have been specified in the Menu Items dialog box.
7. Click *OK*.

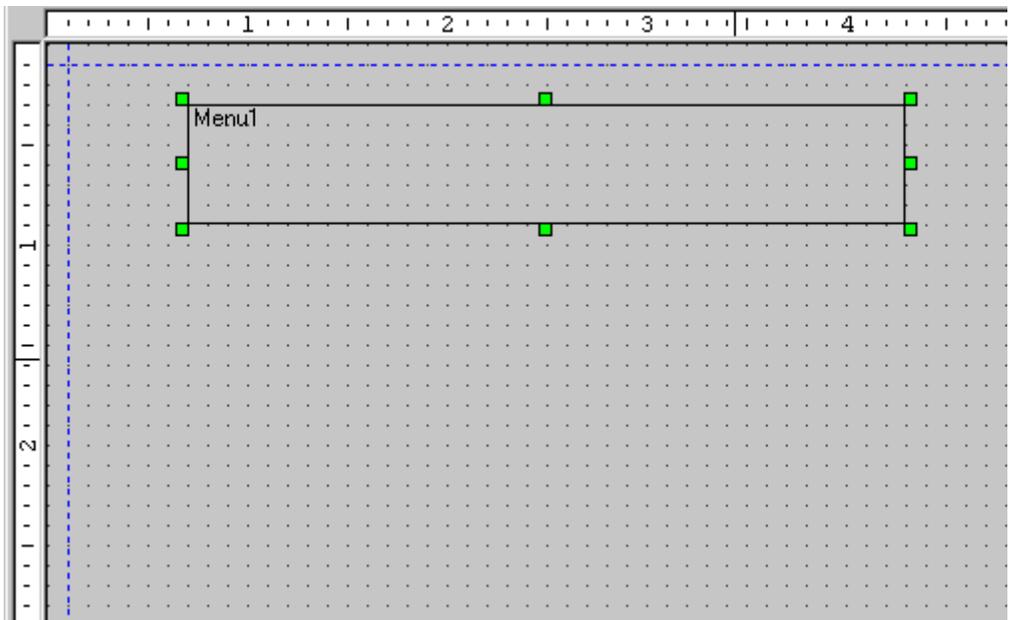
8. Assign a Click action to the lowest level menu items using the Event Handler editor. You can access the Event Handler editor one of the following ways:
  - Right click the menu bar and select *Edit event handlers* from the pop up menu.
  - Select the Events tab in the Properties box and click the *Browse...* button.

For more information on assigning actions to events, see Chapter 5, *Defining Events and Event Handlers*.

### **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.

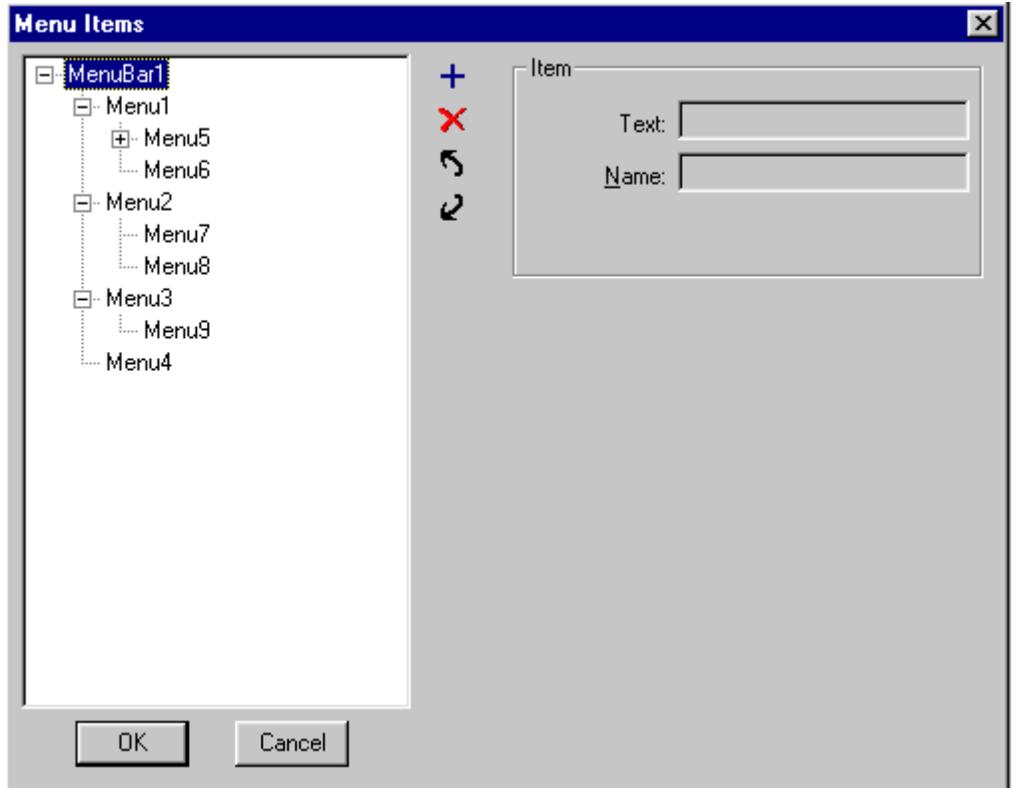
1. Select the *Menu*  control from the Controls Palette.
2. Draw a horizontal rectangle on your form in order to create a horizontal menu bar.



**Note:** You can change the orientation of the menu control by changing the Orientation property in the property sheet.

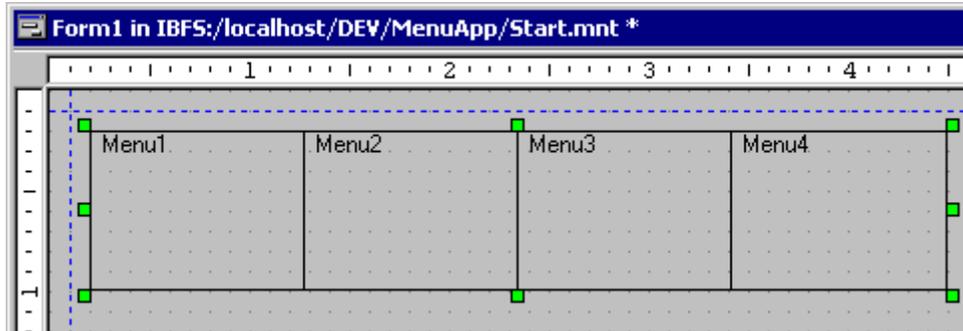
3. Double-click the menu bar to open the Menu Items dialog box.

4. Create menu items on your form by selecting MenuBar1 and clicking the *Add* button. Type the text 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 Add 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*.

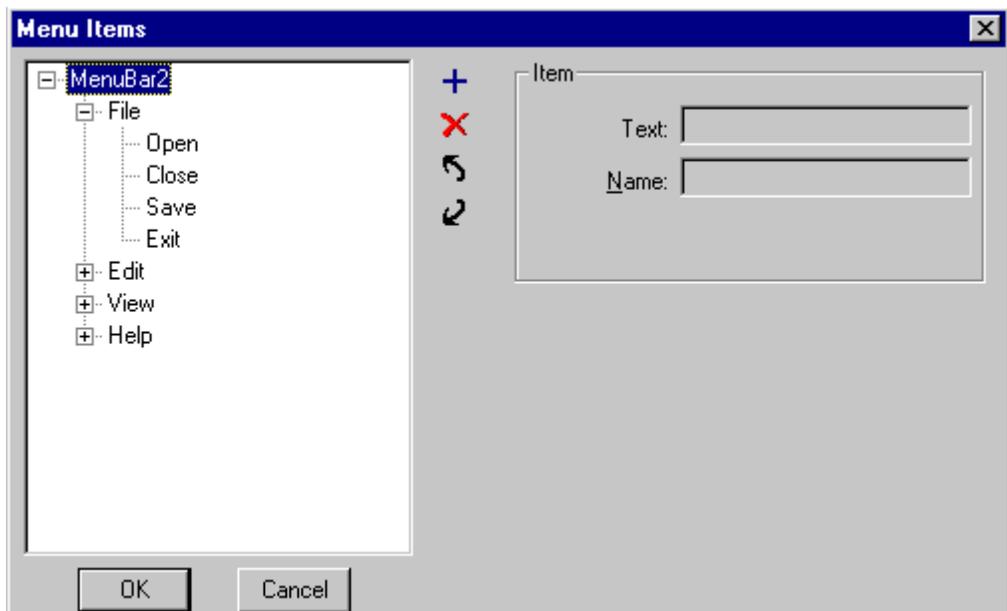
The highest level menu items will appear on your form. Readjust the size and placement of your menu bar if necessary. An arrow on a menu item indicates that the menu item contains submenus.



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 Chapter 5, *Defining Events and Event Handlers*.

### Reference **The Menu Items Dialog Box**

You use the Menu Items dialog box to determine the menu items that will appear on your menu bar, including pull-down menus and sub menus.



The Menu Items dialog box contains the following elements:

**Text**

Allows you to specify the prompt text for each item.

**Name**

Allows 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 run time.

**Do you want to change the name of the menu bar that identifies it to the procedure?**

Change the (Name) property.

**Do you want to change the color of the menu bar?**

The BackColor property determines what the color of the box is. The ForeColor property determines the color of the text in the box.

**Do you want to add a border to the menu bar?**

The Border property determines whether you have a border.

**Do you want to add a border to an individual menu item?**

The ItemBorder property determines whether you have a border around each menu item.

**Do you want to change the size or location of the menu bar?**

Change the Bottom, Left, Right, and Top properties. (You can also move or resize the menu bar directly in the form.)

**Do you want to change what the cursor looks like when it is on top of the menu bar?**

Use the `CursorPointer` property.

**Do 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` and `ForeColorOver` properties.

**Do you want to make the menu bar inactive or make it invisible?**

The `Enabled` property determines whether the menu bar is active or not. (If the menu bar is inactive, it will be grayed out and nothing will happen when the end user clicks it.) The `Visible` property determines whether the menu bar is visible to the end user.

**Do 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 6-110.

**Do you want to move the menu bar to another layer?**

Use the `Layer` property. For more information, see *Layering Controls* on page 6-12.

**Do you want to change the alignment of your menu bar from vertical to horizontal, or vice versa?**

Use the `Orientation` property.

**Do you want to display a tool tip when the cursor is on top of the menu controls?**

Use the `ToolTipText` property.

## 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 6-26. 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 6-29.

## **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. (Optional, but recommended.) Give your radio buttons a more meaningful name than `RadioButtonn`.
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. You 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 6-24.
  - **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* in Chapter 2, *Developing Procedures*.)
  - **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* in Chapter 2, *Developing Procedures*.)
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. You 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* in Chapter 2, *Developing Procedures* or *How to Create a Variable in a Procedure* in Chapter 2, *Developing Procedures*.)
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 the radio button control is placed on.

*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* in Chapter 4, *Developing and Using Forms*.

**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.

### **Do you want to change the contents of the radio buttons?**

Use the ListItems property to open the List Source dialog box.

### **Do you want to change the name of the radio button group that identifies it to the procedure?**

Change the (Name) property.

### **Do 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.

### **Do you want to change the size or location of the radio buttons?**

Change the Bottom, Left, Right, and Top properties. (You can also move or resize the radio button group directly in the form.)

### **Do you want to change the color of the radio buttons?**

The BackColor property determines what the color of the box is. The ForeColor property determines the color of the text.

### **Do you want to change the font in the radio buttons?**

Use the Font property.

### **Do you want to add a label to the radio buttons?**

Turn on the Border property, and assign the label to the BorderText property.

### **Do you want to add a border to the radio buttons?**

The Border property determines whether you have a border. The BorderColor property determines the color of the border. The BorderWidth property determines the width of the border.

### **Do you want to change the number of columns or rows displayed?**

Use the Columns or Rows properties.

**Note:** The size of the radio button group also determines how many rows or columns you see.

### **Do you want to make the radio buttons inactive or make them invisible?**

The Enabled property determines whether the radio buttons are active. (If the radio buttons are inactive, they will be grayed out and nothing will happen when the end user clicks them.) The Visible property determines whether the radio buttons are visible to the end user.

**Do you want to change what the cursor looks like when it is on top of the radio buttons?**

Use the CursorPointer property.

**Do you want to display a tool tip when the cursor is on top of the radio buttons?**

Use the ToolTipText property.

**Do 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 6-110.

**Do you want the end user to be able to tab to the radio buttons?**

Use the Tabstop property.

**Do you want to move the radio buttons to another layer?**

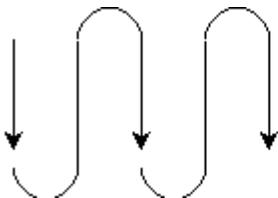
Use the Layer property. For more information, see *Layering Controls* on page 6-12.

## 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.



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 will also need to 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.

### **Do you want to change the text?**

Change the Text property. (You can also select the text and change it directly on the form.)

### **Do you want to change the name of the text that identifies it to the procedure?**

Change the (Name) property.

### **Do you want to change the size or location of the text control?**

Change the Bottom, Left, Right, and Top properties. (You can also move or resize the text directly in the form.)

### **Do you want to change the color of the text?**

The BackColor property determines what color the background is. The ForeColor property determines the color of the text. If your text control has an event handler assigned to its Click event, the Hyperlink property uses the end user's browser settings to determine the color of the text.

### **Do you want to change the text font?**

Use the Font property.

### **Do you want to change the text alignment (left-justified, centered, right-justified)?**

Use the Alignment property.

### **Do you want to add a border to the text?**

The Border property determines whether you have a border. The BorderColor property determines the color of the border. The BorderWidth property determines the width of the border.

### **Do you want to make the text inactive or make it invisible?**

The Enabled property determines whether the text is active or not. (If the text is inactive, it will be grayed out and nothing will happen when the end user clicks it.) The Visible property determines whether the text is visible to the end user.

### **Do you want to change what the cursor looks like when it is on top of the text?**

Use the CursorPointer property.

### **Do you want to display a tool tip when the cursor is on top of the text?**

Use the ToolTipText property.

**Do 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 6-110.

**Do you want to move the text to another layer?**

Use the Layer property. For more information, see *Layering Controls* on page 6-12.

## Using Text Controls as Hyperlinks

One common use for text controls is as 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. You need to 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 4, in the View menu, click *Internet Options...*, then click the Security tab, select *Custom*, and click *Settings...* 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 approximately 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 6-101. 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. (Optional, but recommended.) Give your ActiveX control a more meaningful name than the default one.
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 have no 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, you will need to 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* in Chapter 5, *Defining Events and Event Handlers*.

```
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**

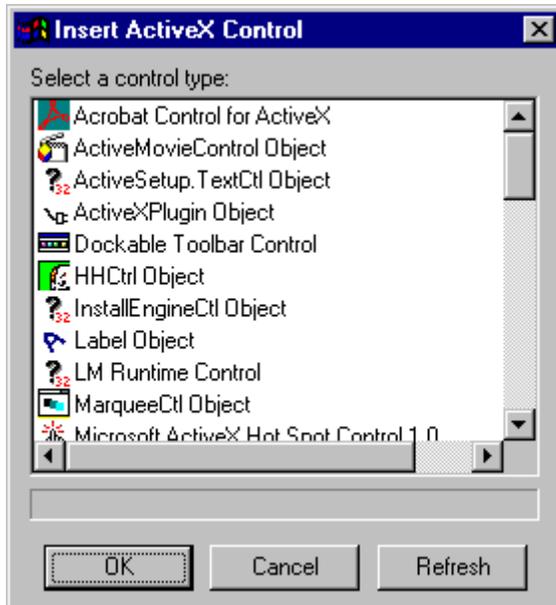
### **Passing an ActiveX Control Value to a Maintain Function**

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, via either JavaScript or VBScript:

```
IWCTrigger ("UpdateDate", document.Form1.CalendarControl.Month);
```

**Reference Insert ActiveX Control Dialog Box**

You use the Insert ActiveX Control dialog box to select an ActiveX control to include as a resource in your project.



This dialog box contains the following elements:

**Select a Control Type:**

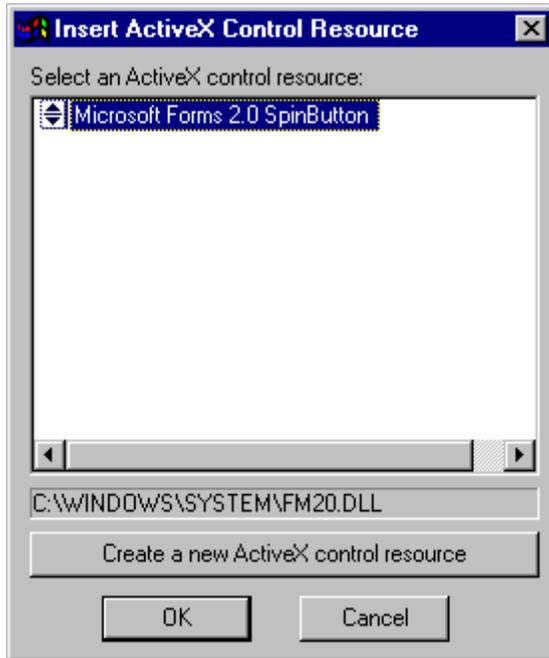
Contains a list of the supported ActiveX controls available on your machine.

**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

You 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.

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.

### **Do you want to change the name of the ActiveX control that identifies it to the procedure?**

Change the (Name) property.

### **Do you want to change the size or location of the ActiveX control?**

Change the Bottom, Left, Right, and Top properties. (You can also move or resize the ActiveX control directly in the form.)

### **Do you want to make the ActiveX control inactive or make it invisible?**

The Enabled property determines whether the ActiveX control is active or not. (If the ActiveX control is inactive, it will be grayed out and nothing will happen when the end user clicks it.) The Visible property determines whether the ActiveX control is visible to the end user.

### **Do you want to change what the cursor looks like when it is on top of the ActiveX control?**

Use the CursorPointer property.

### **Do you want to display a tool tip when the cursor is on top of the ActiveX control?**

Use the ToolTipText property.

### **Do 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 6-110.

### **Do you want the end user to be able to tab to the ActiveX control?**

Use the Tabstop property.

### **Do you want to move the ActiveX control to another layer?**

Use the Layer property. For more information, see *Layering Controls* on page 6-12.

## **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. You can do the following:

- Determine the value of a control property.
- Set the value of a control property.

### ***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.

*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* in *WebFOCUS Maintain Language Reference*.

**Tip:** You can easily 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 information on how to set color properties, see *Defining Colors for Your Form and Controls* on page 6-108.

## **Defining Global Attributes**

---

Some form and control attributes are shared among many or all of the controls, including:

- Defining colors.
- Assigning help HTML files to forms and controls.
- Assigning tab order to controls.

## 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. You use the Color well to assign colors to all of these properties.

Following is a list of properties that assign colors. For more information on these properties, see *Form and Control Properties Reference* in *WebFOCUS Maintain Language Reference*.

- `AlternateRowColor` makes alternate rows in an HTML Table or Grid different colors.
- `BackColor` determines the color of the form or control.
- `BorderColor` determines the color of a control's borders.
- `ForeColor` determines the color of text in a control.
- `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. If you change either of these properties for the form, then this change will be reflected in all of the controls on the form that use the Default setting.

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 pre-defined colors or define your own custom color.

### **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  
Click *Other...* to select one of the basic colors.
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

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

```
[WIFORM SHOW_INACTIVE form;]
form.[control.]SETcolorproperty(amount_of_red, amount_of_green,
amount_of_blue);
```

where:

*WIFORM 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 on *Red*, *Green*, or *Blue*.

**Tip:** You can easily 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, but every once in a while you need to give your end users a little nudge in the right direction.

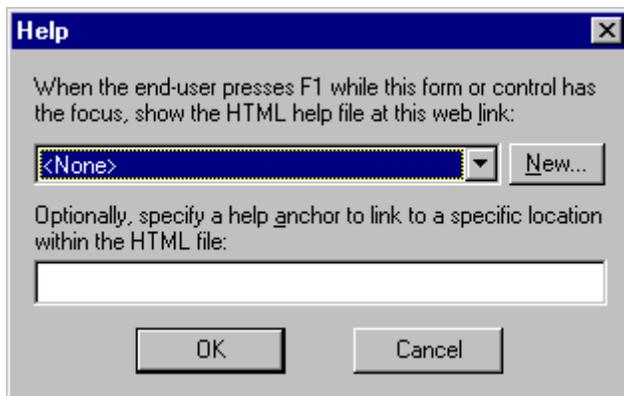
Using the Help property, you can assign a Web link to a form or control. If your Web link is an HTML file, then you can also assign an anchor that identifies a location in the HTML file. Then, when the end user presses F1 in the form or while focus is on the control, the Web link opens.

### **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 information on defining a Web link, see *How to Create a Web Link* in Chapter 5, *Defining Events and Event Handlers*.)
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

You use the Help dialog box 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.



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. See *How to Create a Web Link* in Chapter 5, *Defining Events and Event Handlers* for more information.

**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.

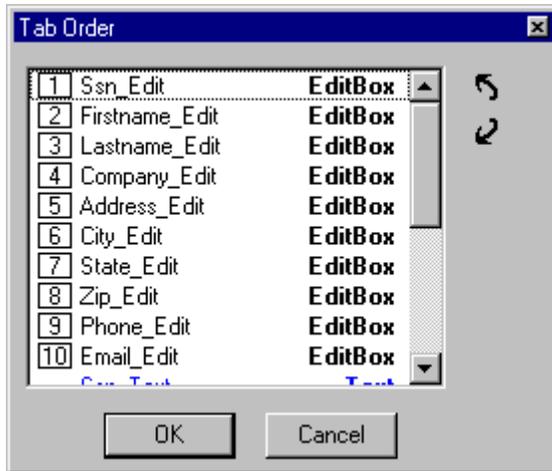


3. Click *OK* to save your changes.

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**

You use the Tab Order dialog box to determine what order an end user pressing Tab in your form will go in.



This dialog box has the following elements:

### **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.

---

---

## CHAPTER 7

# Developing Classes and Objects

### Topics:

- What Are Classes and Objects?
- Defining Classes
- Reusing Classes: Class Libraries
- Declaring Objects

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. A customer object's data might include the customer's ID code, phone number, and order history; the customer's processes might include a function that adds the customer to a new mailing list, a function that updates the customer's contact information, and a function that places an order for the customer.

Object-oriented development—because it models the real-world objects with which your enterprise deals, and encourages you to reuse application logic in a variety of ways—is a more efficient way of developing applications. WebFOCUS Maintain enables you to create applications using object-oriented development, procedural development, or a hybrid of these two methods, providing you with a flexible development path.

## 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's 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 class's variables. 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.)

A class's member functions define its behavior—that is, 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 class's member variables might include the customer's code, name, phone number, and most recent order number:

```
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 could assign a value to Frances Smith's 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's member functions might include a function that adds the customer to a new mailing list, a function that updates the customer's 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 ...
CASE PlaceOrder ...
ENDDESCRIBE
```

After declaring a new customer object for the customer Frances Smith

```
DECLARE CustFrSmith/Customer;
```

you could add Frances Smith to the mailing list using the AddToList member function:

```
CustFrSmith.AddToList();
```

Each object has the same member functions, and so 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's properties—that is, it inherits all of the superclass's member variables and member functions. 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 7-14.)
- **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 Text 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 7-6.)

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 class's member variables. A class's member variables 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* in Chapter 2, *Developing Procedures*.

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 class's member functions. A class's member functions define the actions that can be performed on the class's objects.

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* in Chapter 2, *Developing Procedures*.

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 class's member variables. A class's member variables express its properties. A subclass inherits all of its superclass's member variables, 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* in Chapter 2, *Developing Procedures*.

Note that 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 class's member functions. A class's member functions define the actions that can be performed on the class's objects.

Note that you cannot delete member functions inherited from the superclass; however, you can override an inherited member function 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* in Chapter 2, *Developing Procedures*.

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 Text 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.

```
DESCRIBE classname = ( [superclass +] memvar/type [, memvar/type] ...)
[;]

[memfunction
[memfunction]...
ENDESCRIBE]
```

where:

*classname*

Is the name of the class that you are defining. The name is subject to the Maintain language's standard naming rules; see *Specifying Names* in *Language Rules Reference* in *WebFOCUS Maintain Language Reference* for more information.

*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 class's member variables. The name is subject to the Maintain language's standard naming rules; see *Specifying Names* in *Language Rules Reference* in *WebFOCUS Maintain Language Reference* for more information.

*type*

Is a data type (a built-in format or a class).

*memfunction*

Defines one of the class's member functions. Member functions are defined the same way as other Maintain functions, using the CASE command; see *Command Reference in WebFOCUS Maintain Language Reference* for more information.

;

Terminates the definition if the definition omits member functions. If it includes member functions, the semicolon is omitted and the ENDDESCRIBE command is required.

ENDDESCRIBE

Ends the class definition if it includes member functions. If it omits member functions, the ENDDESCRIBE 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* in Chapter 2, *Developing Procedures*. For instructions on creating variables, see *How to Create a Variable in a Procedure* in Chapter 2, *Developing Procedures*.

To edit one of the class's member functions or member variables:

1. Right-click one of the class's 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* in Chapter 2, *Developing Procedures*; for information about editing functions, see *How to Edit a Function* in Chapter 2, *Developing Procedures*. For general information about editing a class definition, see *Defining Classes* on page 7-4.

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 Text 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 `ENDDESCRIBE`. For more information about the `DESCRIBE` command, see *Command Reference in WebFOCUS Maintain Language Reference*.
4. Close the Text 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 F2.

or

Click the class, member variable, or member function twice.

2. Type the new name.
3. Press Enter 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

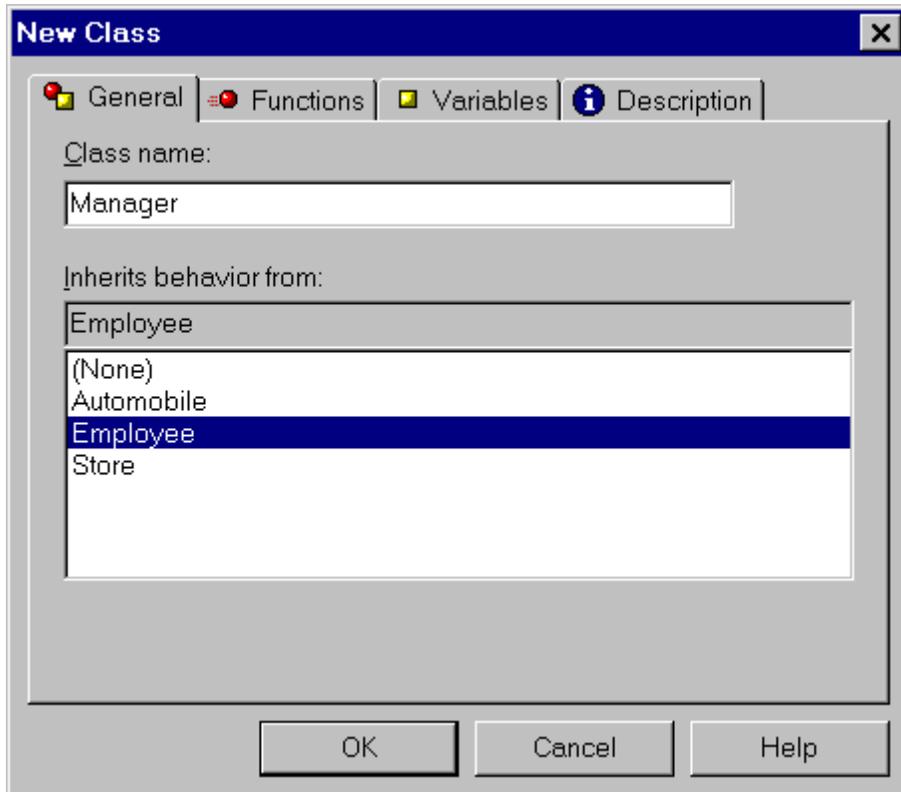
Press the Delete key.

or

Click the *Delete*  button on the General toolbar.

## Reference **New Class Dialog Box: General Tab**

The New Class dialog box enables you to create classes.



The General tab has the following options:

### **Class name**

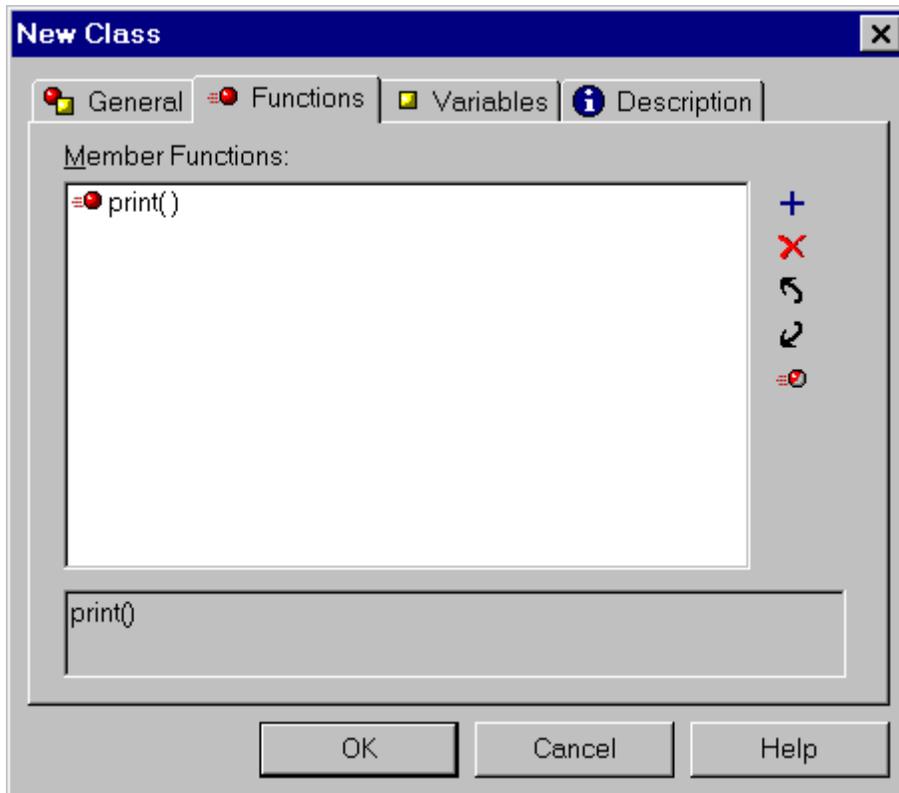
Type the name of your class here. For information about names see *Specifying Names* in *Language Rules Reference* in *WebFOCUS Maintain Language Reference*.

### **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 dialog box enables you to create classes.



The Functions tab has the following elements:

### **Member Functions**

Lists the names of the class's member functions.



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.



Moves a selected function up in the list of functions.



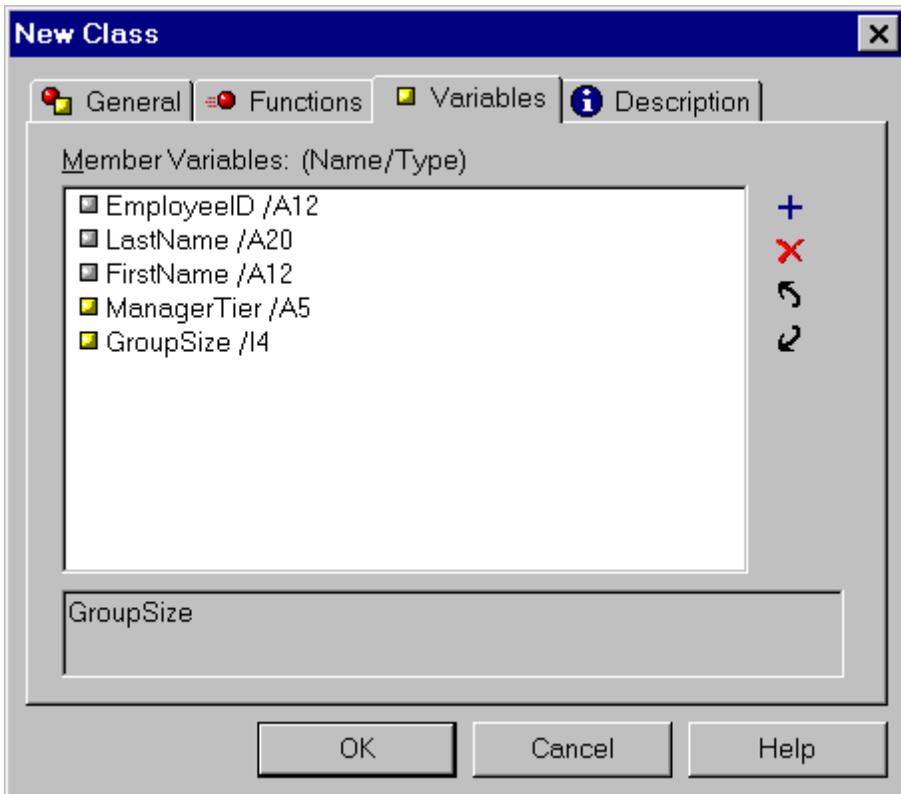
Moves a selected function down in the list of functions.



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 dialog box enables you to create classes.



The Variables tab has the following elements:

**Member Variables: (Name/Type)**

Lists the class's member variables, including those that it has inherited.



Opens the Member Variable dialog box in which 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.



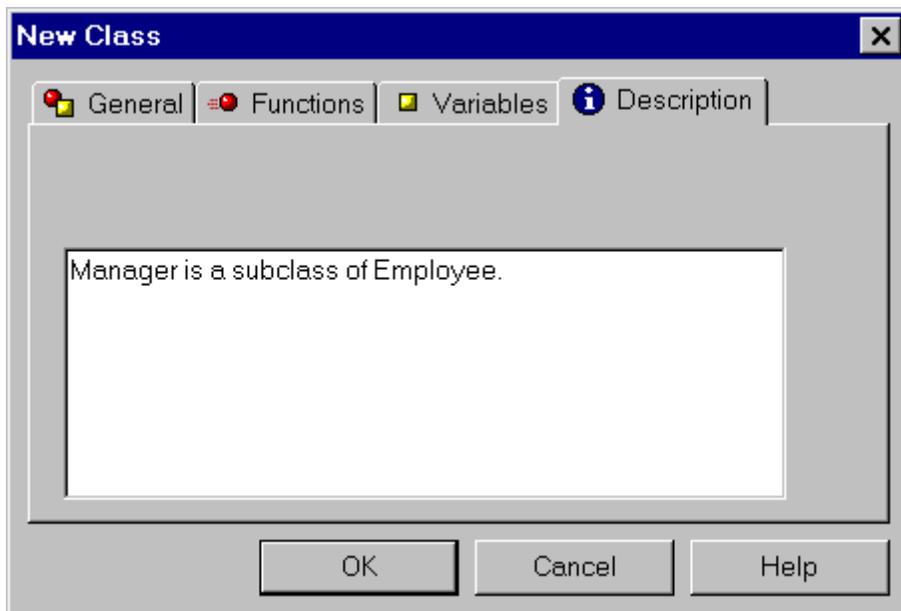
Moves a selected variable up in the list of variables.



Moves a selected variable down in the list of variables.

**Reference New Class and Edit Class Dialog Boxes: Description Tab**

The New Class dialog box enables you to create classes.



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's 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). Once 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* in Chapter 2, *Developing Procedures*. Once 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 7-4.

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 that 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

```
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 procedure's MAINTAIN command.

## Declaring Objects

---

Once 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 Text 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's 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  button to open the Type Wizard and continue with Step 5. See *Using the Type Wizard* in Chapter 2, *Developing Procedures*.
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 Text 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 Text 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 *Command Reference* in *WebFOCUS Maintain Language Reference*.)

To declare an object in the Text Editor using the DECLARE command, use this syntax

```
DECLARE  
[ (  
objectname/class;  
.  
.  
.  
[ ) ]
```

where:

*objectname*

Is the name of the object that you are creating. The name is subject to the Maintain language's standard naming rules; see *Specifying Names* in *Language Rules Reference* in *WebFOCUS Maintain Language Reference* for more information.

*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.

---

---

## CHAPTER 8

# Debugging WebFOCUS Maintain Applications

### Topics:

- Type on EDAPRINT
- Maintain Statement Trace

WebFOCUS Maintain contains two tools that help you debug your applications: Type on EDAPRINT and Maintain Statement trace.

## 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, filename, 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 runtime:

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. The level of detail in the trace output at runtime depends on the type of compile statement used.

- A basic compile command, such as a MNTCON COMPILE procname statement, will cause trace lines to be issued for each Case, Endcase and Call and Exec statement.
- A DEBUG option, available at either compile time as well as a server setting and an execution command, will generate trace output for every Maintain statement.

You can set Compiling and Compiling with Debug at deployment time from within Developer Studio with settings in the Property option of a Deploy Scenario folder. For more information, see the *WebFOCUS Developing Reporting Applications* 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
```

## Maintain Statement Trace

```
12.59.30 FU MNTSTMT>MNTCAR1      Line= 101.008 F= 1  S=CaseDef
006 Msec=46770750 Cpu= 0 Mmem= 2052 Amem= 0

12.59.30 FU MNTSTMT>MNTCAR1      Line= 14.006 F= 2  S=CaseDef
006 Msec=46770750 Cpu= 0 Mmem= 2052 Amem= 0

12.59.31 FU MNTSTMT>MNTGET1      Line= 5.006 F= 1  S=CaseDef
006 Msec=46771691 Cpu= 0 Mmem= 2315 Amem= 0

12.59.31 FU MNTSTMT>MNTGET1      Line= 11.001 F= 1  S=CaseEnd
001 Msec=46771751 Cpu= 0 Mmem= 2316 Amem= 0

12.59.31 FU MNTSTMT>MNTCAR1      Line= 17.019 F= 2  S=CallStmt
005 Msec=46771761 Cpu= 0 Mmem= 2254 Amem= 0

12.59.31 FU MNTSTMT>MNTCAR1      Line= 18.001 F= 2  S=CaseEnd
001 Msec=46771761 Cpu= 0 Mmem= 2254 Amem= 0

12.59.31 FU MNTSTMT>MNTCAR1      Line= 103.003 F= 1  S=CaseEnd
001 Msec=46771761 Cpu= 0 Mmem= 2254 Amem= 0

12.59.31 FU MNTSTMT>MNTCAR1      Line= 132.003 F= 1  S=CaseEnd
001 Msec=46771761 Cpu= 0 Mmem= 2254 Amem= 0
```

### **Example** MNTCON DEBUG EX MNTCAR1 using MNTSTMT

Note lines for Maintain statements Reposition, Stack Clear, Next and For statements:

```
13.21.17 FU MNTSTMT>MNTCAR1      Line= 109.008 F= 1  S=CaseDef
006 Msec=48077990 Cpu= 0 Mmem= 2052 Amem= 0

13.21.17 FU MNTSTMT>MNTCAR1      Line= 120.003 F= 1  S=CaseEnd
001 Msec=48077990 Cpu= 0 Mmem= 2052 Amem= 0

13.21.17 FU MNTSTMT>MNTCAR1      Line= 122.008 F= 1  S=CaseDef
006 Msec=48077990 Cpu= 0 Mmem= 2052 Amem= 0

13.21.17 FU MNTSTMT>MNTCAR1      Line= 101.008 F= 1  S=CaseDef
006 Msec=48077990 Cpu= 0 Mmem= 2052 Amem= 0

13.21.17 FU MNTSTMT>MNTCAR1      Line= 14.006 F= 2  S=CaseDef
006 Msec=48077990 Cpu= 0 Mmem= 2052 Amem= 0

13.21.19 FU MNTSTMT>MNTGET1      Line= 5.006 F= 1  S=CaseDef
006 Msec=48079672 Cpu= 0 Mmem= 2315 Amem= 0

13.21.19 FU MNTSTMT>MNTGET1      Line= 8.012 F= 1  S=RepoStmt
031 Msec=48079672 Cpu= 0 Mmem= 2315 Amem= 0

13.21.19 FU MNTSTMT>MNTGET1      Line= 9.013 F= 1  S=StakClear
035 Msec=48079672 Cpu= 0 Mmem= 2315 Amem= 0

13.21.19 FU MNTSTMT>MNTGET1      Line= 10.040 F= 1  S=NextStmt
017 Msec=48079682 Cpu= 0 Mmem= 2315 Amem= 0
```

```

13.21.19 FU MNTSTMT>MNTGET1      Line=   10.040 F= 1  S=NextStmt
017 Msec=48079682 Cpu=    0 Mmem=  2315 Amem=    0

13.21.19 FU MNTSTMT>MNTGET1      Line=   10.040 F= 1  S=NextStmt
017 Msec=48079682 Cpu=    0 Mmem=  2316 Amem=    0

13.21.19 FU MNTSTMT>MNTGET1      Line=   10.040 F= 1  S=NextStmt
017 Msec=48079682 Cpu=    0 Mmem=  2316 Amem=    0

13.21.19 FU MNTSTMT>MNTGET1      Line=   10.040 F= 1  S=MntForSt
015 Msec=48079682 Cpu=    0 Mmem=  2316 Amem=    0

```

## 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:

- From the server Web Console.
- 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 *Command Reference* in *WebFOCUS Maintain Language Reference*.

Refer to the WebFOCUS Developer Studio documentation for your server platform for information on setting traces from the Web Console and for profile commands.

### **Procedure** How to Set Server Traces on From the Server Web Console

To set server traces on:

1. Select *Diagnostics*, then *Traces*.
2. Click *Enable Traces*.
3. To set Maintain statement traces, click *Configuration*.
4. Select the *Custom Components* radio button.
5. Select *MNTSTMT* (with or without *MNTPERF* for agent memory and CPU) under *Maintain Programmer Traces*.

### **Example** Enabling Server Tracing in EDASPROF.PRF

The following commands enable server tracing, disable all trace levels except MNTSTMT:

```

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:

```
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

```
FILEDEF MNTSTMT DISK E:\IBI\APPS\MY52APP\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 like the following:

TOD	PROCNAME	STMTTYPE	CPUMSEC	MNTMEM	AGNMEM
---	-----	-----	-----	-----	-----
13.28.08	MNTNEW1	CaseDef	11010	2315	11016
		CaseEnd	11010	2316	11016
		MntForSt	11010	2316	11016
		NextStmt	11010	2316	11016
			11010	2316	11016
			11010	2316	11016
			11010	2316	11016
			11010	2316	11016
		RepoStmt	11010	2316	11016
		StakClear	11010	2316	11016

### Sample Usage Scenarios for Tracing

- Problem:** During 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.

Note that once 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.

*Maintain Statement Trace*

---

---

## CHAPTER 9

# Developing an Application for a Shared Application Server

### Topics:

- Why Use a Shared Application Server?
- Writing Applications for the Shared Application Server
- Deploying and Testing Applications on Shared Application Servers
- Preparing a Shared Application Server

The Shared Application Server is a dedicated configuration of an WebFOCUS Maintain Server in pooled mode and is designed for running high-performance highly-scalable applications. The server allows you to widen an application's availability while retaining its efficiency, enabling it to perform well for both small and large groups of users. For example, you could develop an application for a department, and then scale it up to the entire enterprise.

## Why Use a Shared Application Server?

---

By running an application on a Shared Application Server, you can increase the number of its end users while maintaining effective performance with regard to:

- **Response time.** End-users get an immediate response when starting an application because the application is pre-loaded and compiled. Once in the application, users will not be bottle necked by another user's transaction, because the server's application guidelines call for small transactions and ensure that a user walking away from an open form will not lock a data source.
- **Memory.** The average additional user consumes a minimum amount of additional memory because that user shares the server's agent with other users.

You can install the Shared Application Server as a separate WebFOCUS Server or choose to configure an alternate service on an existing WebFOCUS Server to run a Shared Application Server partition.

For information on installing and configuring a Shared Application Server, see the *WebFOCUS Developer Studio Installation and Configuration* manual.

If you wish to optimize a server instance's configuration, you can do so using the Web Console; the console is described in your WebFOCUS Developer Studio documentation. For information about how to optimize your server, contact your Information Builders representative.

To run an application on a Shared Application Server, you must first develop and deploy it in accordance with the server's guidelines, as described in the following topics.

## Writing Applications for the Shared Application Server

---

Ensure that the application's logic conforms to the following guidelines:

- **Stacks.** We recommend reading only as many rows into a stack as are needed for the current transaction, and then clearing the stack when its data is no longer needed, to ensure maximum memory availability. You can specify which rows to read into a stack by using the NEXT command's FOR and WHERE phrases, 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 an end user's current transaction before displaying a form to an end user. You must ensure the transaction's integrity; see *Designing Transactions for the Shared Application Server* on page 9-3.

- **DBA Settings, SYS\_MGR commands.** When using DBA settings, or any SYS\_MGR commands in your procedures, be aware that when your application issues a WINFORM SHOW command, an end user's session is temporarily suspended while other connections are being processed. During this time, your settings could be changed!

Therefore, you should reset your values immediately after the WINFORM SHOW command.

Keep in mind the following limitations:

- User and group server profiles are not supported. Use global profiles only.
- For maximum scalability, and to minimize the memory footprint of the Shared Application Server, we recommend deploying only one Maintain application for each Shared Application Server instance.
- Applications written to update FOCUS data sources require the use of the FOCUS Database Server (SU).

## **Designing Transactions for the Shared Application Server**

An application written for a Shared Application Server achieves its performance benefits, in part, by sharing each server agent among several application users. Each user has sole use of the agent for the duration of a logical transaction; when one user's transaction closes (by being committed or rolled back), the next application user gets sole use of the agent for the duration of his or her logical transaction.

You can maximize performance and minimize bottlenecks by accounting for this sharing behavior: design your application's logical transactions to be as small as possible, so that the server agent's resources rotate frequently to each user.

But what would happen if an application were to display a form requiring a user response, and the user walked away from the workstation, causing the logical transaction to pause until he or she returned and responded to the form? The application's other users on that server agent would be paused indefinitely, unable to proceed until that first user continued.

Maintain avoids this situation by automatically issuing an implied ROLLBACK command just before the application displays a form. This closes the current transaction, ensuring that while the current user is looking at the form (or is away from the workstation), the next user's transaction can exploit the server agent's resources. This ensures high application throughput.

Of course, you will probably want to define your application's transaction boundaries yourself, and not have them defined by this automatic rollback prior to each form. You have two options:

- **Supply your own change-verify logic.** You can code your own change-verify logic to verify, before writing each record, that it has not been changed by other users since you had initially read it for the current transaction. This requires locking each record while it is being retrieved, releasing the lock, working with the data, and then relocking it shortly before writing it to the data source. This technique ensures transaction integrity by verifying, before writing a record, that it has not been changed by other users in the interim.

For more information about this technique, see Chapter 11, *Ensuring Transaction Integrity*. Although the technique is described there for use with DB2 data sources, you can apply it to transactions against other kinds of data sources, changing DBMS-specific details when necessary.

- **Display no forms within a transaction.** You can choose to display all forms before, or after, a transaction's logic, but not in the middle of its logic. This works well for applications that can separate presentation logic from transaction logic. For example, you may have a batch update application that displays some menu forms to enable a user to select different options, and then begins a transaction using data from a transaction file.

## Deploying and Testing Applications on Shared Application Servers

---

An application written for the Shared Application Server should always first be deployed to and tested in a private WebFOCUS Server Environment.

A pooled SAS service is dedicated to running Maintain applications, and will not accept the deployment of application files from Developer Studio. You can deploy to one of the following:

- A private service of a WebFOCUS Server that also has a separate pooled service for SAS.
- A separate instance of a private WebFOCUS Server used for development.

For more information on configuring for SAS refer to the *WebFOCUS Developer Studio Installation and Configuration* manual.

## Preparing a Shared Application Server

---

Before you run an application on a Shared Application Server instance, you must append the following commands to the end of the server instance's global profile or service profile

```
MNTCON PREPARESERVER
-INCLUDE loadproc
MNTCON STARTSERVER
```

where:

*loadproc*

Is the name of an external procedure that contains MNTCON LOADIMAGE commands for the application you wish to run. The name must be uppercase.

The external procedure must reside in the server instance's EDASYNR path.

The syntax of an application's MNTCON LOADIMAGE commands is

```
MNTCON LOADIMAGE [dirname1/]procl
.
.
.
MNTCON LOADIMAGE [dirnamen/]procn
```

where:

*dirname1* / *dirnamen*

Are the names of the directories on the server where you can find the application's procedures.

*procl* / *procn*

Are the names of all of the application's Maintain procedures. The names must be uppercase.

Alternatively, you can place the MNTCON LOADIMAGE commands directly in the profile in place of the -INCLUDE statement. However, using the -INCLUDE statement makes it easier for you to switch a server instance from one application to another.

### **Example** Switching an SAS Server Instance Between Different Applications

For example, the following commands switch a server instance between the AcctPay, Courses, and Sales applications. The server instance is currently set up to run the AcctPay application:

```
MNTCON PREPARESERVER
-INCLUDE ACCTPAY
-* -INCLUDE COURSES
-* -INCLUDE SALES
MNTCON STARTSERVER
```

**Example** **EDASPROF Setting Up a Shared Application Server to Use SU on FOCUS Data Sources**

The following is a sample EDASPROF.PRF for a Shared Application Server using SU for FOCUS data sources:

```
APP ENABLE
APP PATH ACCOUNTS
-*
USE
NYACCTS ON FOCUSU01
REGIONS ON FOCUSU01
END
SET COMMIT=ON
-*
MNTCON PREPARESERVER
MNTCON LOADIMAGE NYACCTS
.
.
.
MNTCON STARTSERVER
```

To run the starting procedure NYACCTS, you would use the MNTCON RUNIMAGE NYACCTS command from your launch form.

---

---

## CHAPTER 10

# Managing Team Development

**Topic:**

- Team Development

This topic outlines the ways in which multiple users can develop a WebFOCUS Maintain application. While this release of WebFOCUS Maintain does not fully support team development, it does allow for sharing components of a project as described in this topic.

## Team Development

---

WebFOCUS Maintain will fully support team development, including having developers check in and check out shared application components, in a future release. While the current release of WebFOCUS Maintain does not fully support team development, it does enable multiple users to collaboratively develop a WebFOCUS Maintain project in the following way:

1. **Create a version of the project.** Create your own version of the shared project. For example, if you are one of four developers working together to create an application named Account, you would create a project named Account on your local development server and so would all the other developers.

You would then work directly in your version of the project on the components for which you are responsible.

2. **Define a team deployment scenario.** Everyone on the development team should define a common deployment scenario; this ensures that all of the project's components will be available to everyone on the team. This team deployment scenario should deploy all project components to WebFOCUS Servers (not locally). Each procedure in the shared application should have only one owner; that is, each procedure should be a native component in only one developer's version of the application. For information about deployment scenarios and deploying applications, see *Partitioning and Deploying Project Files* in *WebFOCUS Developing Reporting Applications*. For information about team deployment scenarios for applications designed for Shared Application Servers, see Chapter 9, *Developing an Application for a Shared Application Server*.

On the deployment server, everyone on the team would use the same user ID (for example, TeamID) which ensures that they will all be using the same user profile (TeamID.prf) and therefore the same server parameter settings. This approach applies to all the WebFOCUS Servers in the team's deployment scenario. For information about server parameters and profiles, see *WebFOCUS Developer Studio Installation and Configuration* manual.

3. **Create and deploy your initial work.** Create basic versions of the procedures for which you are responsible, and deploy them using the team's deployment scenario. This makes the procedures available to the rest of the team, who can then link your deployed procedures into their own versions of the project.

Each procedure in the shared application should have only one owner; that is, each procedure should be a native component in only one developer's version of the application.

- 4. Deploy and run locally for private testing.** When testing your work, consider first deploying your version of the project locally, to Local Development, to test your changes independently of everyone else's changes. Testing may require local data.
- 5. Deploy and run for team testing.** When you are sufficiently confident of your changes and think that they are ready to incorporate into the team's latest "build" of the application, deploy your version of the project using the team's deployment scenario. Your changes will then become available to the rest of the team. (For information about deploying and running applications designed for Shared Application Servers, see Chapter 9, *Developing an Application for a Shared Application Server*.

You will be running the deployed application to test your work in the context of the latest source code that each developer has deployed.

- 6. Edit locally and repeat the process.** After team-testing your application, edit it locally as necessary, then repeat steps 5-7 as necessary.



---

---

## CHAPTER 11

# Ensuring Transaction Integrity

### Topics:

- Why Is Transaction Integrity Important?
- Defining a Transaction
- Evaluating If a Transaction Was Successful
- Concurrent Transaction Processing
- Ensuring Transaction Integrity for FOCUS Data Sources
- Ensuring Transaction Integrity for DB2 Data Sources

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.)

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; see your DBMS vendor’s 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 11-10. 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 11-18. For many other types of data sources, you can also apply the strategies described in *Ensuring Transaction Integrity for DB2 Data Sources* on page 11-18, 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:

- 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.** Your data sources' database management system (DBMS) may require that you perform some tasks to enable transaction integrity; see your DBMS vendor's documentation for information. You can set some native DBMS parameters using the `SYS_MGR.FOCSET` command; see *Command Reference* in *WebFOCUS Maintain Language Reference*. You can also set some native DBMS parameters through FOCUS and iWay; see your iWay documentation.

For FOCUS data sources, you need to 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 11-10.

- 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's boundary. For more information see *Defining a Transaction* on page 11-3.
- 4. Evaluate the transaction's success.** 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 If a Transaction Was Successful* on page 11-8.

## 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 all-or-nothing 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”, unbalancing the bank’s 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 transaction’s last data source command. (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 transaction’s boundary. 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’s boundary, as described in *When an Application Ends With an Open Transaction* on page 11-8.)

### **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:

```
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—attempting to update a record that doesn't exist—you might wish to ask the application end user to correctly re-enter the customer code (which is being used as the record's key). 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 transaction's success or failure.

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. If you ever need to cancel an open transaction, you must do so by issuing a ROLLBACK command. ROLLBACK voids any of the transaction's data source commands 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's procedures. (Resetting a data source's position markers points them to the beginning of the data source's 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

```
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 11-9.

## Designing Transactions That Span Procedures

Logical transactions can span multiple Maintain procedures. If a Maintain procedure with an open transaction passes control to an external 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 transaction's boundaries 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 child's open transaction** 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. KEEP is described in *Optimizing Performance: Data Continuity and Memory Management in WebFOCUS Maintain Concepts in Getting Started*, and in *Command Reference in WebFOCUS Maintain Language Reference*.
- **You can close the child's open transaction** 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. RESET is described in *Optimizing Performance: Data Continuity and Memory Management in WebFOCUS Maintain Concepts in Getting Started*, and *Command Reference in WebFOCUS Maintain Language Reference*.

**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 application's transaction boundaries—that is, how the choice between KEEP and the implied COMMIT determines where the initial transaction ends, and how many transactions follow.

<b>C returns to B with...</b>	<b>B returns to A with...</b>	<b>Transaction boundaries (    )</b>
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 application's procedures that read and write to a given data source reside on the same WebFOCUS Server.
- In each of the application's transactions 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 If a Transaction Was Successful**

---

When you close a transaction by issuing a COMMIT or ROLLBACK command, you need to determine if 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 need to 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 line that evaluates the success of the transaction is shown in **bold**:

```
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; one transaction's changes 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) needs to coordinate concurrent access. There are many strategies for doing this. No matter which type of data source you use, WebFOCUS Maintain respects your DBMS's 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 vendor's documentation. For FOCUS data sources, this information is presented in *Ensuring Transaction Integrity for FOCUS Data Sources* on page 11-10. 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 11-18. For many other types of data sources, you can also apply the strategies described in *Ensuring Transaction Integrity for DB2 Data Sources* on page 11-18, changing DBMS-specific details when necessary.

### **Example Why Concurrent Access to a Data Source Needs to Be Carefully Managed**

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's 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 Irving's record and, based on information in a transaction data source indicates that she has been promoted to manager, computes her new job code (A16).
2. The Salary application reads Irving's record 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's job code but an assistant manager's salary). 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 as the 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 its 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 11-11.

- **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 11-14.
- **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 11-15.
- **Start the FOCUS Database Server.** Under Windows NT 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*.

## 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 server's group profiles. 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 server's user profiles (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's global profile, or in one or more of its user or group profiles, using the following syntax

```
SET COMMIT={ON|OFF}
```

To set COMMIT in a Maintain procedure, use the following syntax:

```
SYS_MGR.FOCSET ("COMMIT" " {ON|OFF} "
```

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. This is the default.

## 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—in which 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 allows 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 FOCUS Database Server's change-verification strategy 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.  
**Caution:** 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's 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 11-14.)

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 application's FocCurrent transaction variable 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's 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 PATHCHECK server parameter.

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 NT 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 NT and UNIX.

## Identifying the FOCUS Database Server

To identify which FOCUS Database Server will manage access to a given FOCUS data source, you need to issue a USE command that associates the server with the data source.

You can issue the USE 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's 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 need to associate the data source with the server by issuing a USE command in a WebFOCUS Server profile. The USE command's 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 NT and UNIX is the node name of the FOCUS Database Server, as defined in the FOCUS Database Server node block of the iWay 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.

External procedures—such as FOCUS report procedures—are not part of a logical transaction; when control passes from a Maintain procedure to an external procedure, the open transaction is suspended for the duration of the external procedure. Therefore, if the external procedure reports against a FOCUS data source, it accesses the live data source, not the open transaction's private copy. 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 external 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 external 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 11-16. Otherwise, the external 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 external 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 and/or external procedures that are located on the target server.

This requirement also applies to the target server itself: if it executes Maintain procedures and/or external 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 external 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 Accessing Maintain and External Report Procedures on the Same Server**

In an application that uses a FOCUS Database Server, if you want to deploy to the application's Maintain procedures and external 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 *Developing Reporting Applications*.
2. **Represent the target server to WebFOCUS Servers as two outbound nodes.** Perform this step for each WebFOCUS Server that executes Maintain procedures and/or external procedures that are deployed on the target server. This requirement also applies to the target server itself: if it executes Maintain procedures and/or external 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's node block in the WebFOCUS Server's communications configuration file 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's node block from WebFOCUS Developer Studio `odin.cfg` file (located in the `webfocusdesktop_root_dir\srv52\wfs\etc` directory), and paste it, twice, into the target server's communications configuration file. Edit the second pasted block to provide a new eight-character node name, but leave the block's other values unchanged.

For information about the name and location of the server's communications configuration file, see *Communications Configuration File Location and Name* on page 11-18.

3. **Deploy the application.** For information about deploying applications, see *Developing Reporting Applications*.

## **Reference Communications Configuration File Location and Name**

The WebFOCUS Server communications configuration file under:

- **Windows NT and UNIX** is `odin.cfg`, and resides in the `etc` subdirectory of the iWay Data Server configuration directory. Under UNIX, `odin.cfg` is in the `$EDACONF/etc` directory; under Windows NT, 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/srv52/server_instance` under UNIX, and `home\ibi\srv52\server_instance` under Windows NT. `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's 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 a transaction's isolation level; 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 11-6, and in *When an Application Ends With an Open Transaction* on page 11-8.

We recommend two strategies for writing transactions to DB2 data sources:

- **Transaction locking.** 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.
- **Change verification.** 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. (For information about applying change verification—for all types of data sources—to applications designed for Shared Application Servers, see Chapter 9, *Developing an Application for a Shared Application Server*.)

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's DB2 Logic Differs From Other IBI Products

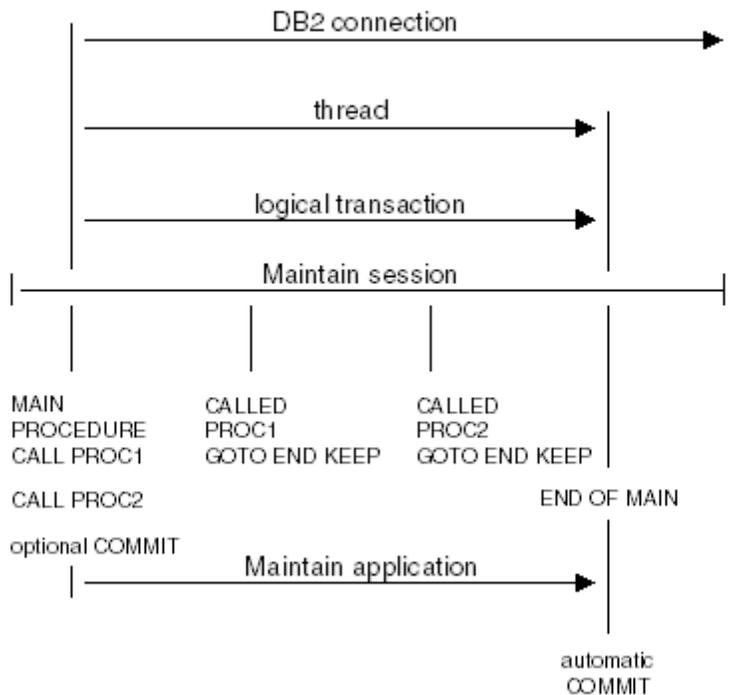
If you are familiar with using the DB2 Data Adapter with Information Builders products other than WebFOCUS Maintain, note that WebFOCUS Maintain works with DB2 a bit differently:

- 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 11-6, and in *When an Application Ends With an Open Transaction*.
- Maintain does not support the FOCUS and iWay command SQL DB2 SET AUTOCOMMIT to control automatic commits.
- Because Maintain works on sets of rows, the DB2 Data Adapter does not automatically generate change verification logic.

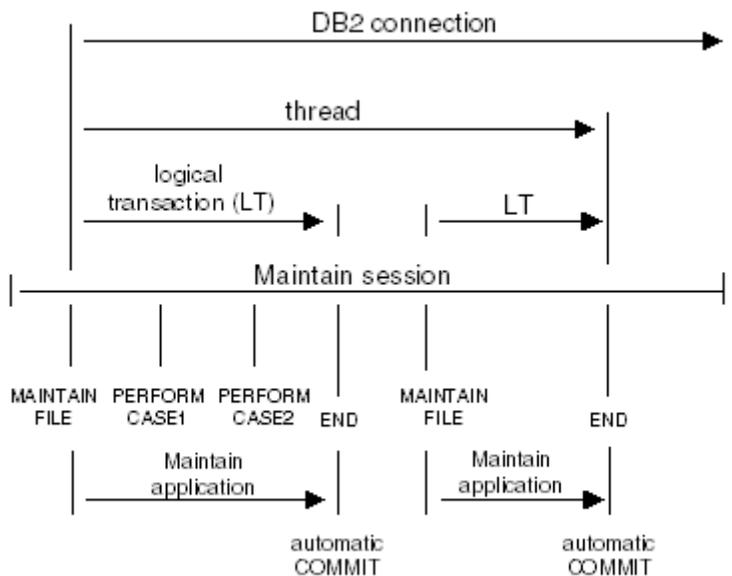
## Using Transaction Locking to Manage DB2 Row Locks

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 the change verification strategy, 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 11-22.
- 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 transaction's updates 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 11-6.

**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 via 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 11-22.

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.

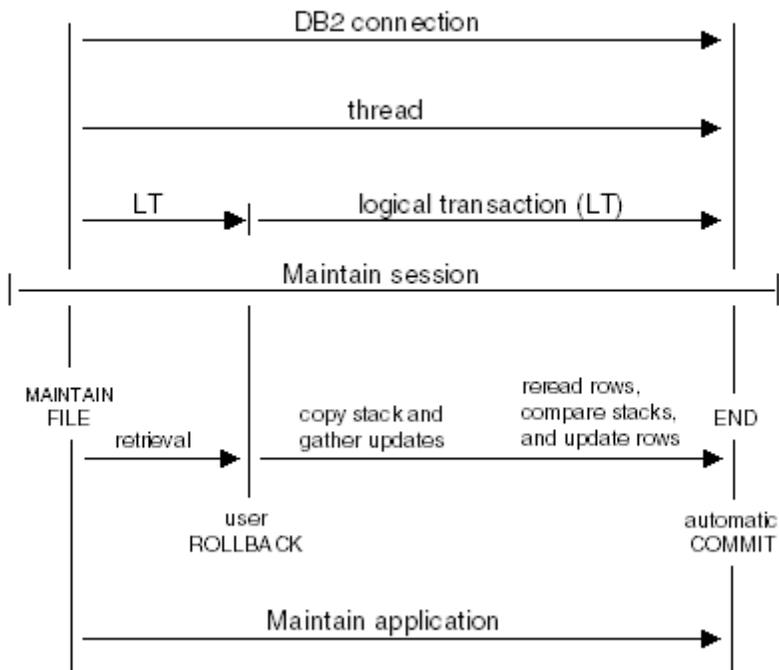
## Using Change Verification to Manage DB2 Row Locks

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. For information about applying change verification—for all types of data sources—to applications designed for Shared Application Servers, see Chapter 9, *Developing an Application for a Shared Application Server*.

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, you retrieve the original rows from the data source again, relocking them in the process. You 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 the transaction locking strategy, 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 transaction's updates to the rows in the original stack (Stack1).
- 5. Re-read the rows.** Re-retrieve the transaction's rows 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 (that is, Stack2) to the current data source values (that is, 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's updates to the data source, and issue a COMMIT command to commit the transaction.

The COMMIT or ROLLBACK command releases all row locks.

---

---

## CHAPTER 12

# Editing Project Components as Text

### Topics:

- Opening a Project Component
- Typing Text
- Performing Basic Editing Functions
- Finding and Replacing Text
- Sizing the Editor Window
- Changing Colors, Fonts, and Tabs
- Using Bookmarks to Move Within the Editor
- Previewing and Printing Text
- Using the Language Wizard

The Procedure Editor, the Script Editor, and the Event Handler editor are tools that allow you to edit a project component as text. This topic includes information on how to edit a project component and how to use the Language Wizard.

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 External Procedure Editor enables you to edit the code for an external procedure.
- 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. Changes you make to code in the editor are parsed and automatically incorporated in the corresponding graphical application component when you leave the editor window.

When you open a procedure, external procedure, script, or HTML file in the editor window, syntax elements in the text appear color-coded for easy viewing and editing. For example, the keywords in a procedure will appear in blue. You can change default colors or remove text coloring to suit your preferences. See *Changing Text Color* on page 12-9 for details.

**Tip:** For help about syntax for keywords in the Maintain language, highlight the keyword in the Procedure Editor window and press F1. You will see topics from *WebFOCUS Maintain Language Reference* that provide the information you need.

The Procedure Editor, the Script Editor, and the Event Handler editor parse your Maintain language or script code for errors. If they find an error, they display an exclamation point in the margin next to the code with the error. You will also see information about the error in the General tab of the Output window. You can easily open the editor at the line with the error by double-clicking the syntax in the Output window.

The editor enables you to use familiar Windows editing techniques, such as cut, copy, paste, undo/redo, and drag-and-drop. See *Performing Basic Editing Functions* on page 12-3.

In addition you can:

- Bookmark lines for quick editing and easy reference. See *Using Bookmarks to Move Within the Editor* on page 12-10.
- Find and replace text. See *Finding and Replacing Text* on page 12-6.
- Call the Language Wizard to help you create Maintain language code. See *Using the Language Wizard* on page 12-12.

The General and Editor toolbars provide quick access to the most common functions.

## Opening a Project Component

---

To view or edit a procedure or other component as text, you must open it in the editor window. If the component you want to view extends beyond the frame of the editor window, you can use the mouse, the arrow keys on the keyboard, or the mouse wheel (if you have one) to scroll.

You can open several editor windows at the same time and move between them as you work. For example, you can drag-and-drop text between open 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.

## Typing Text

---

You can begin to type text when you open a project component. A blinking bar indicates where your text will appear. You can place the cursor anywhere. When Insert is on, the text is adjusted to fit that line. If you want to replace text, use Overtyping. Because the editor does not wrap text automatically, as you reach the end of a line you must press Enter.

### **Procedure** How to Toggle Between Insert and Overtyping Modes

On the keyboard, press the Insert key to toggle between insert and overtype modes.

Overtyping mode is indicated by the letters OVR on the Maintain Development Environment's status bar; Insert mode (the default) is indicated as blank space.

## Performing Basic Editing Functions

---

If you want to edit a project component, you can use the editor to open it as text and start typing. In addition, the editor has some standard Windows editing features that will expedite your work.

### **Procedure** How to Select a 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.

The text is highlighted.

**Procedure** **How to Select All Text**

To select all the text in the editor, do one of the following:

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

**Procedure** **How to Cut Text**

1. Select the text to cut.
2. In the Edit menu, click *Cut*.

or

Click the *Cut*  button on the General toolbar.

The text you selected has now been moved to the Windows clipboard.

**Procedure** **How to Copy Text**

1. Select the text to copy.
2. In the Edit menu, click *Copy*.

or

Click the *Copy*  button on the General toolbar.

The text you selected has now been copied to the Windows clipboard.

**Tip:** You can copy selected text by holding down the Ctrl key as you drag the text to the desired location.

**Procedure** **How to Paste Text**

1. Position the cursor where you want to paste the text that you cut or copied.
2. In the Edit menu, click *Paste*.

or

Click the *Paste*  button on the General toolbar.

The text you selected has now been pasted.

**Note:** You can only paste the text that was last copied.

**Tip:** If you press the Insert key on the keyboard, activating overtyping (shows as OVR in the status bar), you can replace the entire contents of a window with the new text.

**Procedure How to Delete Text**

1. Select the text to delete.
2. On the keyboard, press Delete.

The text you selected has now been deleted.

**Procedure How to Drag-and-Drop Text Between Editor Windows**

1. Select the text to drag-and-drop into a new location within the same window or in another editor window.
2. Press and hold the left mouse button until the mouse arrow has a rectangle shape at the line tip.
3. Drag the text to the desired location and release the left mouse button.

The text is dropped in its new location.

**Procedure How to Drag-and-Drop Between Other WebFOCUS Maintain Windows and the Editor Window**

1. Select any component in the Project Explorer.
2. Drag the selected text or component to the desired location in the editor window.

The name of the component is dropped in its new location. If the component is a function WebFOCUS Maintain generates the code to execute the function (`PERFORM function;`). If the component is a form, WebFOCUS Maintain generates the code to display the form (`WINFORM SHOW form;`).

**Procedure How to Undo/Redo Text**

To undo or redo an editing action, do one of the following:

- In the Edit menu, click *Undo* or *Redo*.
- Click the *Undo*   or *Redo*   buttons on the General toolbar.

The last action you performed has now been reversed if you selected Undo or repeated if you selected Redo.

**Note:** Multiple undo and redo options become available when more than one edit action is performed.

## Finding and Replacing Text

---

Find enables you to search a project component for specified text. You can also include special characters—for example, paragraph marks and symbols—in your search criteria. Replace lets you search for a word and replace it with another.

Options available to narrow or speed your search are:

- **Match Case.** Finds only those occurrences with the exact combination of uppercase and lowercase letters specified in the Find What box.
- **Regular Expression.** Searches for text using regular expression syntax.
- **Wrap around search.** Searches the entire project component source code from the current insertion point.
- **Direction (to search).** “Up” searches from the insertion point to the beginning of the document. “Down” searches from the insertion point to the end of the document.
- **Mark All.** Marks all of the lines with instances of your search text with a bookmark. For more information, see *Using Bookmarks to Move Within the Editor* on page 12-10.

**Tip:** You can search through your entire project for a text string using the *Search project* command on the Edit menu. The results are displayed in the Find tab of the Output window.

### **Procedure** How to Find Text

1. On the Edit menu, select *Find*.

or

Click the *Find*  button on the Editor toolbar.

The Find dialog box opens.

2. In the Find What box, type the word or phrase you want to find.
3. Tab to and select any search criteria that will help to narrow or speed your search.
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

1. On the Edit menu, select *Replace*.  
The Replace dialog box opens.
2. In the Find What box, type the word or phrase you want to replace.
3. In the Replace With box, type the new word or phrase that you want the old word or phrase to be replaced with.
4. Tab to and select any search criteria that will help to narrow or speed your search.
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.

**Note:** To replace all occurrences of the text found at one time, click Replace All.

## **Sizing the Editor Window**

---

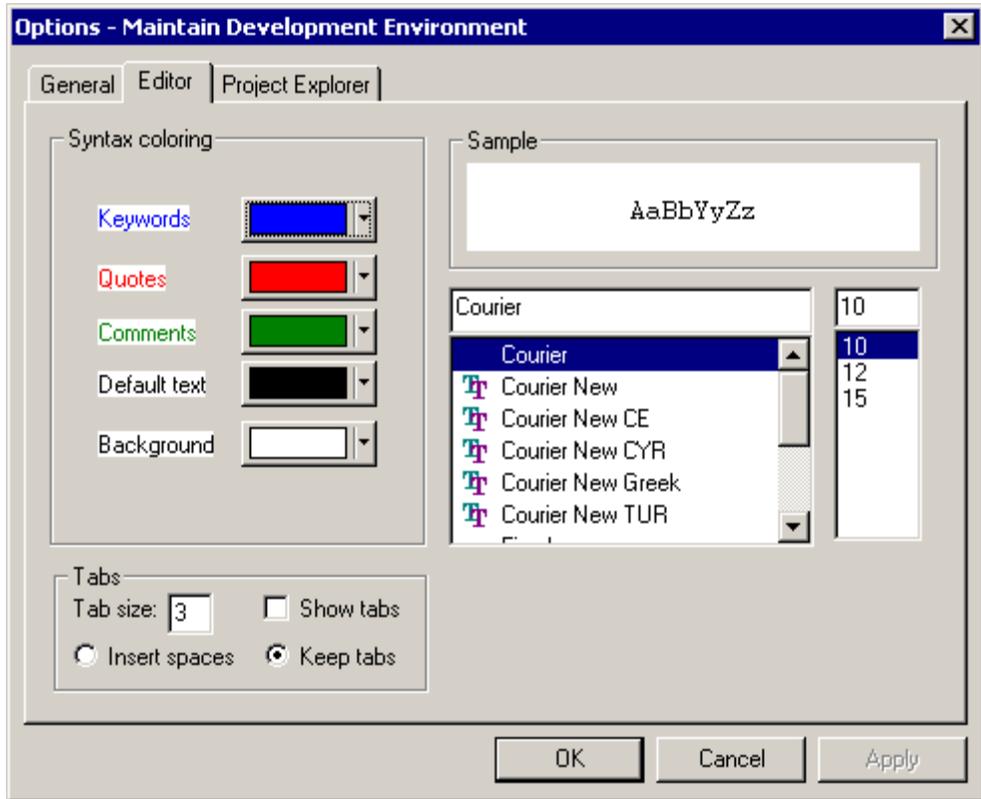
You can use the maximize and restore buttons in the upper-right corner of the editor window to make it smaller so that you can view more than one window at a time, or larger to see more text. If you need more space, you can minimize any editor window to a title bar or remove the Status bar.

## **Changing Colors, Fonts, and Tabs**

---

You can change the colors, font characteristics, and tab settings in the editor using options on the Editor tab. You can access the Editor tab from Environment Options on the Tool menu.

**Reference Editor Tab, Options - Maintain Development Environment Dialog Box**



The Editor tab contains the following fields:

**Syntax Coloring**

Displays default colors for Keywords in the Maintain language, Quotes (text between single or double quotation marks), Comments, Default text, and Background. You can change the color display for any of these elements in the editor window or disable syntax coloring.

**Fonts**

Displays the default font type, size, and style, all of which you can change.

**Tabs**

Displays a default tab size. You can change the tab size, insert spaces, and control whether to keep, show, or hide tabs.

**Sample**

Provides a window in which you can preview your changes before applying them.

**Apply button**

Implements your changes in the editor window without closing the Options dialog box so that you can make additional changes.

**OK button**

Implements your changes in the editor window and closes the Options dialog box.

**Changing Text Color**

You can change the default colors in the editor. This is referred to as syntax coloring.

You can turn syntax coloring off. If you leave it on, you can change the default colors, which include four syntax elements and the background.

Syntax Coloring Options	Description
Keywords	Words in the Maintain language.
Quotes	Any text between a single or double quotation marks.
Comments	Comment characters like the following, and related text: <pre>\$\$ - * \$* . . . * \$</pre>
Default Text	All other text.
Background	The space that all text displays on.

**Procedure How to Change the Text Color**

1. On the Tools menu, select *Environment Options*.
2. Select the *Editor* tab.
3. Select a color for the syntax or window element you want to change.
4. Click *OK*.

**Procedure How to Turn off Syntax Coloring**

1. On the Tools menu, select *Environment Options*.
2. Select the *Editor* tab.
3. Deselect the Use syntax coloring check box.

## Changing the Font Type, Style, and Size

You can set the text to a different font, apply a font style, and change the font size. You can also see a sample of the font before you change it.

### **Procedure** How to Change the Font Type, Style, and Size

1. On the Tools menu, select *Environment Options*.
2. Select the *Editor* tab.
3. Select a font, a font style, and/or a font size.
4. Click *OK*.

## Setting Tabs

You can change the tab settings in the editor to display or hide tabs, change the tab size, insert spaces, and/or keep tabs.

### **Procedure** How to Change Tab Settings

1. On the Tools menu, select *Environment Options*.
2. Select the *Editor* tab.
3. Under the Tabs group, change the Tab size by typing in a new number, show tabs by clicking the *Show tabs* box, or *Insert spaces* or *Keep tabs* by selecting the appropriate radio button.
4. Click *OK*.

The text adjusts to the new tab settings.

**Note:** Insert space and Keep tabs work in conjunction with Tab size.

## 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. Once 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 will disappear when you close the editor window.

### **Procedure** How to Apply or Remove a Bookmark

1. Position the cursor anywhere in a line where you want the bookmark to appear.
2. Click the right mouse button, then select *Toggle Bookmark*.

or

Click the *Toggle Bookmark*  icon:

**Tip:** You can also apply/remove a bookmark by positioning the cursor on a line to be marked, pressing and holding down the Ctrl key while pressing the F2 function key.

### **Procedure** How to Jump to a Bookmark

To jump to another bookmark, click the *Next Bookmark*  or *Previous Bookmark*  icon:

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

### **Procedure** How to Apply Multiple Bookmarks

1. On the Edit menu, select *Find*.  
The Find dialog box opens.
2. In the Find What box, 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.
- Close the editor window and reopen it.

## **Previewing and Printing Text**

---

By previewing a component as text you can see entire pages, or pairs of pages, at reduced size before printing. After previewing the component, you can print it.

### **Procedure** How to Preview an Application Component as Text

In the File menu, click *Print Preview*.

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

**Procedure How to Print an Application Component as Text**

To print, do one of the following:

- In the File menu, click *Print*.
- Click the *Print*  button on the General toolbar.

## Using the Language Wizard

---

WebFOCUS Maintain provides a Language Wizard to help you construct Maintain language source code for the following:

- To retrieve records from a data source (NEXT).
- To update records in a data source (INCLUDE/UPDATE).
- To operate on a forms (WINFORM).
- To run another procedure (CALL or EXEC).
- To operate on a stack (STACK).

You can quickly access the Language Wizard from the Procedure Editor or Event Handler editor.

**Procedure How to Launch the Language Wizard From the Procedure Editor or Event Handler Editor**

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.

---

---

## CHAPTER 13

# Customizing the Maintain Development Environment

### Topics:

- Managing Windows
- Using Toolbars
- Customizing Toolbars
- Using the Status Bar
- Setting Preferences
- Viewing the Tip of the Day

The Maintain Development Environment contains many tools to aid you in your development work:

- You can use standard Windows commands to tile, cascade and split your windows. In addition, several windows are “dockable,” which means they can be anchored to the side of the main window.
- To speed your work, most of WebFOCUS Maintain’s commands are placed on pre-defined toolbars. You can display or hide these toolbars as necessary, or create your own customized toolbars.
- An optional status bar displays information about what WebFOCUS Maintain is doing.
- You can set preferences for how your workspace should be set up.
- You can view a “Tip of the Day” to get hints about how to get things done in WebFOCUS Maintain.
- You can also customize the folders in the Explorer window. For more information on how to do this, see your WebFOCUS Developer Studio documentation.

## Managing Windows

---

As you work in WebFOCUS Maintain, you will open various windows in the Maintain Development Environment, including text editing windows, form editing windows, the output window, tool palettes, and so on. WebFOCUS Maintain supplies standard window management commands to manage these windows.

You can do the following:

- Make a window active.
- Dock and undock windows. Docked windows are aligned to one edge of the Maintain Development Environment window and always appear on top of all other windows. Undocked windows float on the Maintain Development Environment window.
- Split a text editing window into panes. Panes enable you to view different parts of a window simultaneously. It is also very easy to remove them when you are done.
- Tile your windows. WebFOCUS Maintain makes all your undocked windows the same size and arranges them in the available open space in the Maintain Development Environment.
- Cascade your windows. WebFOCUS Maintain makes all your undocked windows the same size, and arranges them on top of each other with only the title bars showing.

WebFOCUS Maintain saves your window positions in the *Project.gfa* file.

### **Procedure** How to Make a Window Active

To make a window active, click its title bar.

If you cannot see the window and you want to make it active, in the Window menu, click its name.

### **Procedure** How to Dock a Window

To dock a window, do one of the following:

- Double-click the window's title bar.
- Drag the window near the edge of the main window.
- Right-click the window, and in the shortcut menu, click *Dockable*.

An outline shows where the window will be docked.

**Note:** You cannot dock a text editing window or a form.

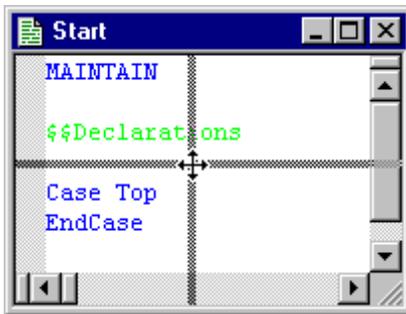
**Procedure How to Undock a Window**

To undock a window, do one of the following:

- Double-click the window's double bar or title bar.
- Drag the window by the double bar to its desired location.
- Right-click the window, and in the shortcut menu, click *Dockable*.

**Procedure How to Split a Text Editing Window**

1. Make sure that the window you want to split is the active window.
2. In the Window menu, click *Split*.
3. WebFOCUS Maintain divides the window into four equal-sized panes with your cursor over the center point.



To split the window horizontally, drag the center point to the left or right side of the window, and click to confirm.

To split the window vertically, drag the center point to the top or bottom of the window, and click to confirm.

**Procedure How to Remove the Split From a Text Editing Window**

1. Place your cursor over the center point of the panes. The cursor will change to indicate that you can adjust the panes.
2. Drag the center point to one of the four corners of the window.

**Procedure How to Tile Windows**

To tile windows, in the Window menu, click *Tile horizontally* or *Tile vertically*.

**Note:** The Tile commands apply only to undocked windows; docked windows are not affected.

### **Procedure** How to Cascade Windows

To cascade your undocked windows, In the Window menu, click *Cascade*.

**Note:** The Cascade command applies only to undocked windows; docked windows are not affected.

## Using Toolbars

---

To help you work more quickly, WebFOCUS Maintain displays toolbars with commonly used commands represented by buttons. To accomplish a function quickly, all you need to do is click the button.

WebFOCUS Maintain ships with built-in toolbars, but you can customize the toolbars. You can:

- Select which toolbars should display.
- Display a toolbar in a moveable window.
- Create toolbars.
- Customize individual toolbars.

### **Reference** WebFOCUS Maintain Built-in Toolbars

WebFOCUS Maintain provides the following built-in toolbars:

- The **General** toolbar contains standard file management commands from the File menu, such as New, Open, Save, and Print, as well as standard Edit menu commands such as Cut, Copy, Paste, Delete, Undo, and Redo.
- The **Editor** toolbar contains commands that help you manage bookmarks in files that you are editing as text, as well as find text.
- The **Maintain Objects** toolbar contains commands create new components in a procedure.
- The **Miscellaneous** toolbar contains commands that display informative information about WebFOCUS Maintain, such as the Properties box, About box, and Tip of the Day, as well as the Help Search dialog box.
- The **Layout** toolbar contains commands that assist you in creating forms in the Form Editor.

If you wish, you can create toolbars.

**Procedure How to Display a Toolbar or Hide a Toolbar**

1. Right-click the toolbar area.
2. In the toolbar shortcut menu, select the toolbar you want to display, or deselect the toolbar you want to remove.

or

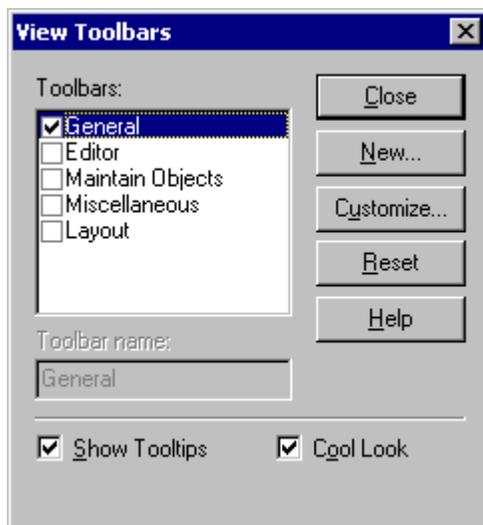
1. In the View menu, click *Toolbars...*
2. In the View Toolbars dialog box, select the toolbar you want to display, or deselect the toolbar you want to remove.
3. Click *Close*.

**Procedure How to Display a Toolbar in a Moveable Window**

By default, toolbars appear in the toolbar area at the top of the screen. If you want your toolbar to appear in a moveable window, double-click the double bar at the left side of the toolbar.

**Reference View Toolbars Dialog Box**

To add or remove toolbars on the window, you can use the View Toolbars dialog box. To open this dialog box, in the View menu, click *Toolbars...*



This dialog box has the following fields:

### **Toolbars**

Displays a list of the toolbars. To view a toolbar in the window, select the check box next to the toolbar. To remove a toolbar from the window, deselect the check box.

### **Toolbar name**

Repeats the name of the toolbar selected in the Toolbars list. If the toolbar is a user-defined toolbar (that is, not one of the built-in ones), you can change the toolbar name.

### **Close**

Closes the dialog box.

### **New...**

Opens the New Toolbar dialog box, where you can create a toolbar. For more information, see *How to Create a Toolbar* on page 13-7.

### **Customize...**

Opens the Customize toolbar dialog box, where you can customize your toolbars. For more information, see *Customizing Toolbars* on page 13-6.

### **Reset**

Returns your settings to the original WebFOCUS Maintain toolbar settings.

### **Show Tooltips**

Turns *tooltips* on and off. Tooltips tell you the name of a button when you place your cursor on the button and leave it there for at least a second. (You do not have to click the button to see the tooltip.)

### **Cool Look**

Turns boxes around the buttons on a toolbar on and off.

## **Customizing Toolbars**

---

If you are unhappy with the standard toolbars, you can create new toolbars that contain the commands you need, using the Customize dialog box. You open the Customize dialog box by clicking *Customize toolbars...* in the Tools menu.

This dialog box has two tabs:

- The Toolbars tab, where you create toolbars.
- The Commands tab, where you determine which commands go on the toolbars.

From here, you can do the following:

- Create toolbars.
- Rename toolbars.
- Delete toolbars.
- Add commands to user-defined toolbars.
- Remove commands from user-defined toolbars.

**Procedure** **How to Create a Toolbar**

1. In the Tools menu, click *Customize toolbars...*  
The Customize dialog box opens at the Toolbars tab.
2. Click *New...*
3. Type a name for your new toolbar.
4. Click *OK*.

Now that you have created a toolbar, you need to add some commands to it. See *How to Add Commands to a Toolbar* on page 13-8.

**Procedure** **How to Rename a User-Defined Toolbar**

1. In the Tools menu, click *Customize toolbars...*  
The Customize dialog box opens at the Toolbars tab.
2. Select the toolbar you want to rename in the list of toolbars.
3. Type a new name in the *Toolbar name* box.
4. Click *OK*.

**Note:** You cannot rename a built-in toolbar.

**Procedure** **How to Delete a User-Defined Toolbar**

1. In the Tools menu, click *Customize toolbars...*  
The Customize dialog box opens at the Toolbars tab.
2. Select the toolbar you want to delete in the list of toolbars.
3. Click *Delete*.
4. Click *OK*.

**Note:** You cannot delete a built-in toolbar.

**Procedure** **How to Add Commands to a Toolbar**

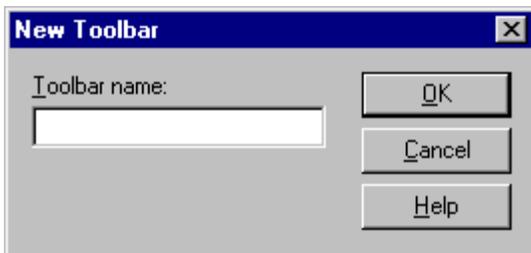
1. In the Tools menu, click *Customize toolbars...*  
The Customize dialog box opens at the Toolbars tab.
2. Click the Commands tab.
3. Make sure that the actual toolbar you want to add commands to is visible.
4. Find the first command you want to add to the toolbar:
  - a. Select a toolbar in the Categories box (the commands are grouped into categories that correspond to the default toolbars).
  - b. Select the command you want to add to your new toolbar. You will see a description of the command in the Description box.
5. Drag the button to your new toolbar in the Maintain Development Environment (the actual toolbar, not the list of categories in the dialog box).
6. Repeat steps 4 and 5 until you have added the commands you wish to the toolbar.
7. Click *OK*.

**Procedure** **How to Remove Commands From a Toolbar**

1. In the Tools menu, click *Customize toolbars...*  
The Customize dialog box opens at the Toolbars tab.
2. Click the *Commands* tab.
3. Make sure that the toolbar you want to remove commands from is visible.
4. Drag the command from the toolbar back to the Customize dialog box.

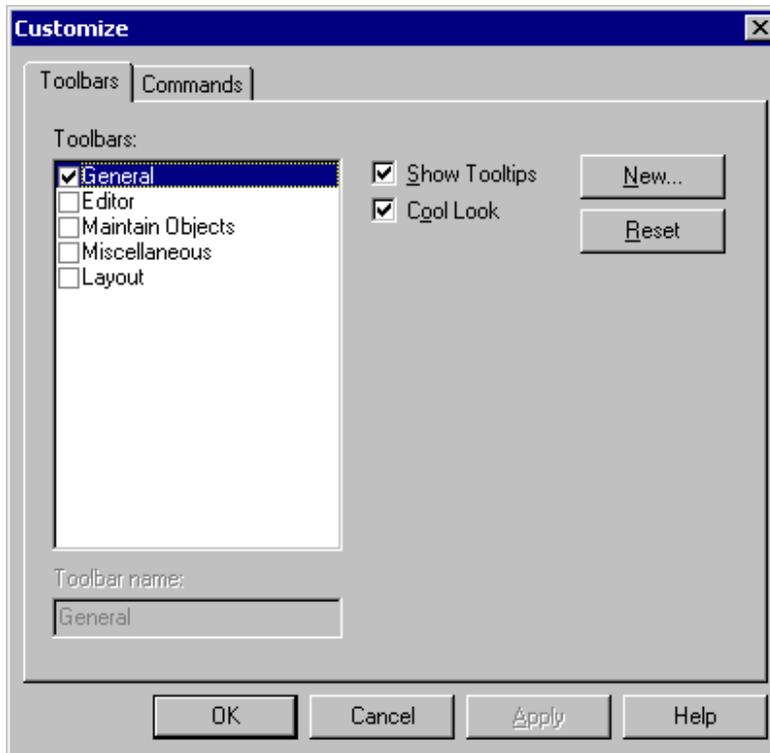
**Reference** **New Toolbar Dialog Box**

You use the New Toolbar dialog box to name toolbars that you create.



## Reference **Customize Dialog Box: Toolbars Tab**

The Toolbars tab in the Customize dialog box is where you create, rename, and delete user-defined toolbars. (To add or remove commands from a user-defined toolbar, see the Commands tab.) To open this dialog box, in the Tools menu, click *Customize toolbars...*



The Toolbars tab has the following fields:

### **Toolbars**

Lists the available toolbars. To make a toolbar visible, turn on the check box to its left.

### **Toolbar name**

Repeats the name of the toolbar selected in the Toolbars list. If the toolbar is a user-defined toolbar (that is, not one of the built-in ones), you can change the toolbar name.

### **Show Tooltips**

Turns *tooltips on and off*. Tooltips tell you the name of the command that WebFOCUS Maintain will execute when you click an icon on a toolbar. They appear near a command on a toolbar when you place your cursor on the icon and leave it there for at least a second.

### Cool Look

Turns boxes around icons on a toolbar on and off.

### New...

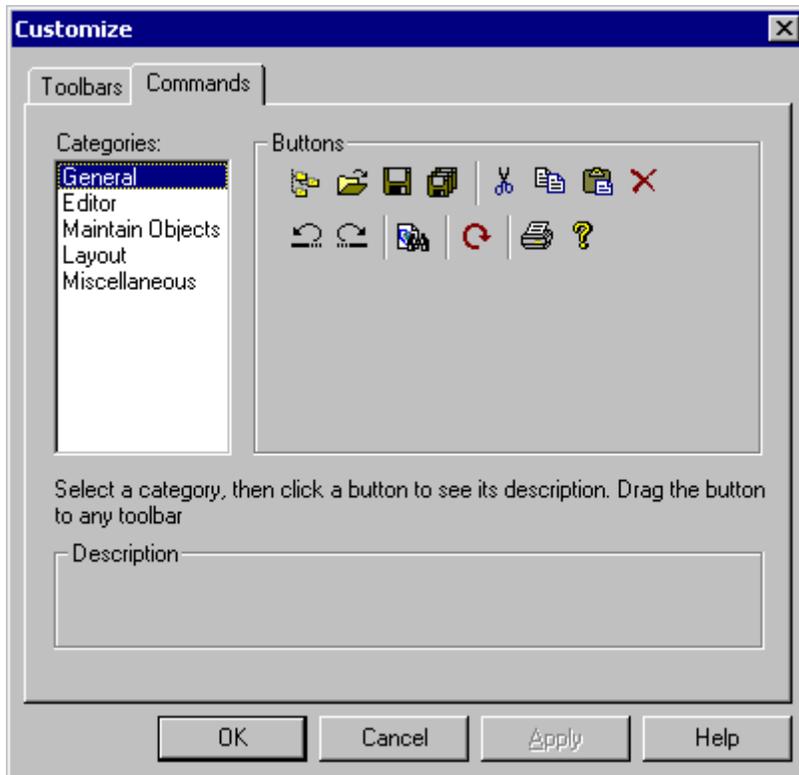
Opens the New Toolbar dialog box, where you can create a toolbar. For more information, see *How to Create a Toolbar* on page 13-7.

### Reset

Returns your settings to the original WebFOCUS Maintain settings.

## Reference **Customize Dialog Box: Commands Tab**

The Commands tab in the Customize dialog box is where you add or remove commands to or from a toolbar. (To create a toolbar, see the Toolbars tab.) To open this dialog box, in the Tools menu, click *Customize toolbars...* and click the *Commands* tab.



This tab displays the following information:

**Categories**

Contains a list of command categories. These categories correspond to the default toolbars. Select a category to see the commands in that category. For more information on the default toolbars, see *WebFOCUS Maintain Built-in Toolbars* on page 13-4.

**Buttons**

Contains icons corresponding to commands in the selected category. Click any icon to see its description. Drag any icon to a toolbar to add it to that toolbar.

**Description**

Contains the description of the selected icon in the Buttons box.

## Using the Status Bar

---

At the bottom of the Maintain Development Environment window is a status bar. The status bar displays the following types of information:

- What WebFOCUS Maintain is doing (Ready for input, deploying a project, running a project).
- If you are in one of the text editors, what line and column your cursor is positioned at.
- Whether Caps Lock is on or off.
- Whether Num Lock is on or off.
- Whether Overtyping mode is on or off.

**Procedure** **How to View or Hide the Status Bar**

In the View menu, click *Status Bar*.

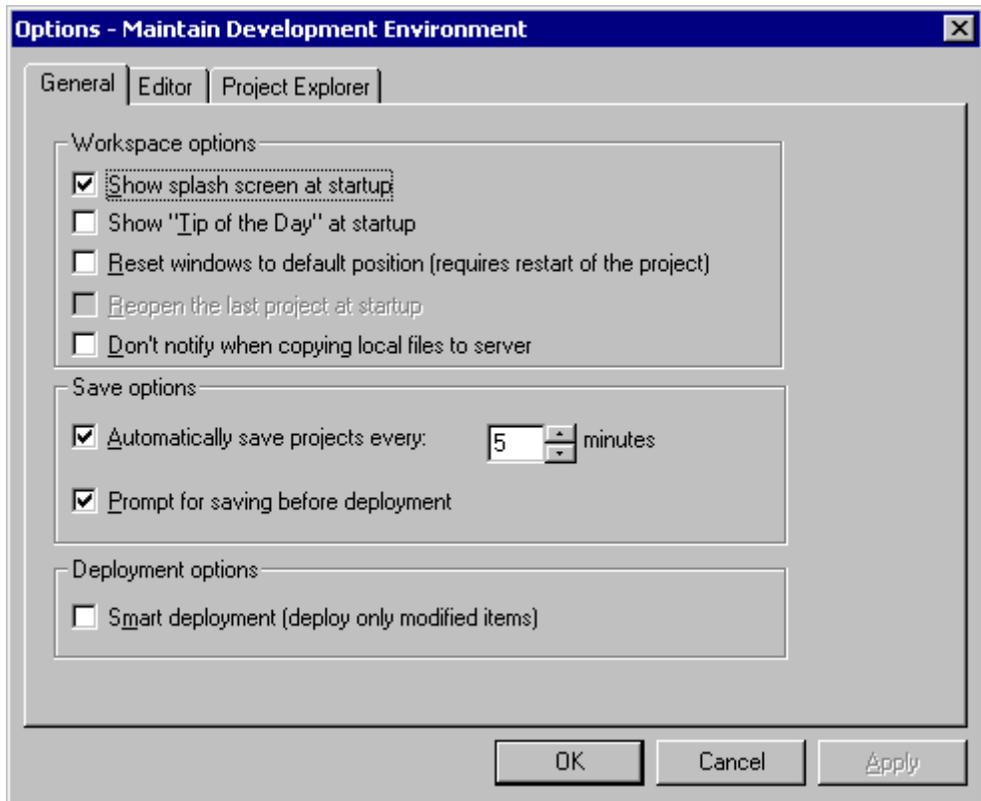
## Setting Preferences

---

WebFOCUS Maintain enables you to set preferences for your environment by choosing *Environment options...* from the Tools menu.

### **Reference** General Tab in Options - Maintain Development Environment Dialog Box

The General tab in the Options - Maintain Development Environment dialog box enables you to set general preferences for the Maintain Development Environment.



This dialog box has the following options:

#### **Show splash screen at startup**

Causes WebFOCUS Maintain to display a splash screen whenever you open the Maintain Development Environment.

#### **Show "Tip of the Day" at startup**

Causes WebFOCUS Maintain to display the Tip of the Day whenever you open the Maintain Development Environment.

**Reset windows to default position (requires restart of the project)**

Causes WebFOCUS Maintain to display windows in their default positions each time you open or restart a project.

**Reopen the last project at startup**

This option is not available when grayed out.

**Don't notify when copying local files to the server**

Allows WebFOCUS Maintain to copy local files to a remote server without notifying you first.

**Automatically save projects every *n* minutes**

Causes WebFOCUS Maintain to save your projects periodically. You can set the interval between saves to any number of minutes you wish.

**Prompt for saving before deployment**

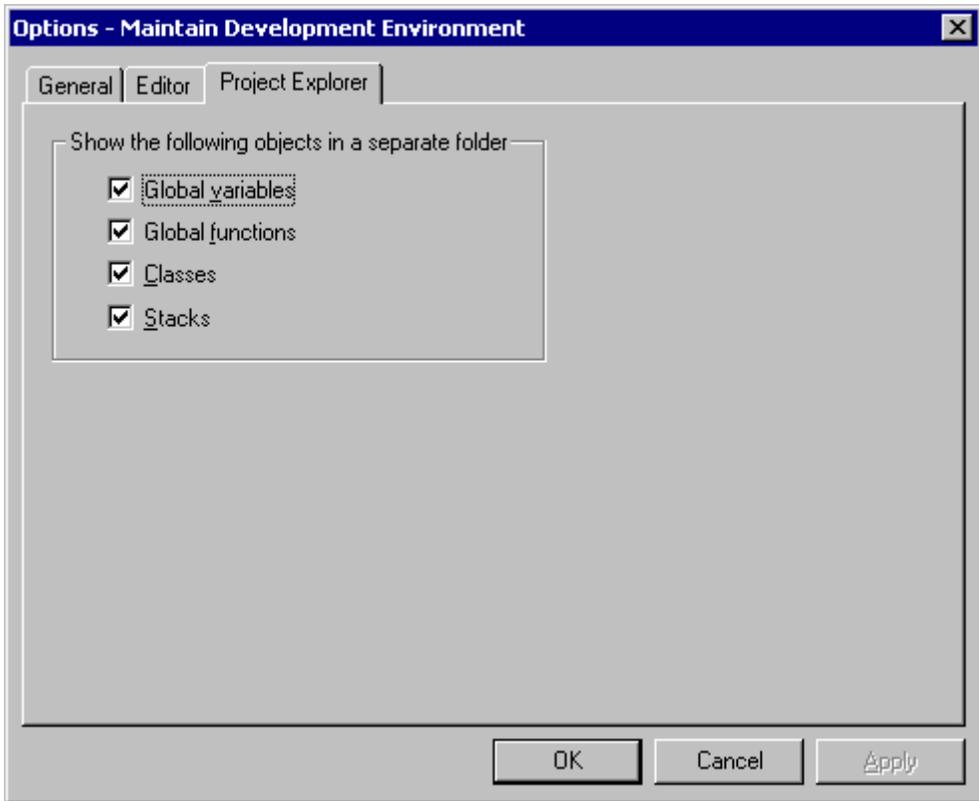
Causes WebFOCUS Maintain to prompt you to save your project before deploying.

**Smart deployment (deploy only modified items)**

When checked, WebFOCUS Maintain deploys only the components of your project that have been modified.

**Reference** **Project Explorer Tab in Options - Maintain Development Environment Dialog Box**

The Project Explorer tab in the Options - Maintain Development Environment dialog box enables you to customize how you display project components in the Project Explorer. You will need to exit and re-enter the Maintain Development Environment in order to see the change.



**Show the following objects in a separate folder**

Controls whether global variables, global functions, classes, or stacks appear under separate folders under a procedure, or whether they appear directly under the procedure.

## Viewing the Tip of the Day

---

When you open WebFOCUS Maintain, you have the option of viewing a “Tip of the Day”. These tips contain useful information about things you can do with WebFOCUS Maintain.

This dialog box contains the following options:

### **Next Tip**

Displays the next tip.

### **Previous Tip**

Displays the previous tip.

### **Show Tips at Startup**

When checked, automatically displays this dialog box at startup.

### **Procedure How to View the Tip of the Day**

1. In the Help menu, click *Tip of the day...*

or

Click the *Tip of the Day* button in the Miscellaneous toolbar.



2. Use *Next Tip* and *Previous Tip* to see other tips.
3. Click *OK* when you are done.

### **Procedure How to Automatically View the Tip of the Day at Startup**

1. In the Tools menu, click *Environment options...*, or in the Help menu, click *Tip of the day...*
2. Turn on *Show Tip [of the day] at startup*.
3. Click *OK*.

*Viewing the Tip of the Day*

---

---

# Index

## Symbols

---

.trc file 8-6

WINFORM command  
SHOW 4-5 to 4-6

## Numerics

---

3GL programs 1-6

## A

---

A0 variables 6-31  
multi-edit boxes and 6-31

Access Files 1-12

active forms 4-7

active windows 13-2

ActiveX controls 6-100, 6-105  
placing in forms 6-100 to 6-101  
unstable 6-106

AddFont function 6-47

AddTab function 6-47

Alert function 5-10 to 5-11

Allow docking command 13-2

Alphanumeric format 2-23 to 2-24

amper variables 5-32  
specifying in URL Wizard 6-81, 6-84

application components  
linking to procedures 10-2  
linking to resources 10-2  
opening as text 12-3  
sharing 10-2

applications 1-2  
closing at run time 4-11, 5-2 to 5-3  
creating 10-2  
debugging 8-1  
driver procedures 4-11 to 4-12  
exiting at run time 4-11  
opening components as text 12-3  
scalable 9-2  
testing 10-2

arguments for functions 2-6

arguments for Maintain procedures 2-26 to 2-27,  
2-29

arguments for procedures 2-28

asterisks next to project name 1-7 to 1-8

AUTOCOMMIT command 11-18 to 11-19

## B

---

background images 6-71

backslash ( character in image paths 6-72

backslash character in image paths 6-59, 6-67,  
6-71

Binding the Selection Result dialog box 6-23, 6-29

blue properties in property sheet 3-5

Blur event 5-5  
in grids 6-43

bookmarks in text editor 12-10  
applying 12-10 to 12-11  
jumping to 12-10 to 12-11  
removing 12-10 to 12-11

Bracket negative edit option 2-23, 2-25

broadcast commit 11-7

built-in toolbars 13-4

- buttons 6-15
  - changing properties 6-17
  - creating 6-15 to 6-16
  - placing on forms 6-15

## C

---

- CALL command
  - in Event Handler editor 5-7
- Caps Lock status 13-11
- Cascade command 13-2, 13-4
- case sensitivity in scripts 5-18
- cases in functions 7-2
- CellGetReadOnly function 6-47
- CellGetTextColor function 6-47
- CellSetReadOnly function 6-47
- Change event 5-5, 6-23, 6-25, 6-93, 6-95
- change verification strategies 11-18
  - for DB2 data sources 11-18, 11-22, 11-24
  - for FOCUS data sources 11-12
- check boxes 6-18
  - changing properties 6-22
  - placing on forms 6-18 to 6-19
  - setting values 6-18 to 6-19
  - triggering actions 6-18, 6-20
- Check event 5-5, 6-18, 6-20
- CHECKED 6-18 to 6-19
- Checked keyword 6-18 to 6-19
- Checked property 6-18, 6-20
- CICS transactions 1-6, 1-14
- Class Editor 7-4, 7-10
  - Description tab 7-4, 7-13
  - Functions tab 7-4, 7-11
- classes 7-1 to 7-2
  - applying 2-23, 2-26
  - defining 7-2, 7-4 to 7-7
  - deleting 7-4, 7-9
  - DESCRIBE command and 7-4, 7-7
  - displaying in a folder 2-4
  - editing 7-4, 7-8 to 7-9
  - importing libraries 7-14
  - inheritance 7-4
  - libraries 7-14
  - member functions 7-2
  - member variables 7-2
  - renaming 7-4, 7-9
  - subclasses and superclasses 7-4, 7-6 to 7-7
- Clear All Bookmarks command 12-10 to 12-11
- Click event 5-5
- ClickArea event 5-5
- ClickColumn parameter 5-10, 5-13, 6-68 to 6-69
- ClickLink event 5-5, 6-68 to 6-69
- ClickRow parameter 5-10, 5-13, 6-68 to 6-69
- clipboards 6-4 to 6-5, 12-3 to 12-4
- Close event 5-5
- CLOSE keyword in WINFORM command 4-7 to 4-8
- CLOSE\_ALL keyword in WINFORM command 4-7, 4-9
- closing a form at run time 5-2 to 5-3
- closing application at run time 5-2 to 5-3
- COBOL programs 1-6
- color properties 6-108 to 6-109
  - setting dynamically 6-108 to 6-109
- colors 6-108
  - applying in Form Editor 6-108
  - customizing in Form Editor 6-108
  - in text editor 12-9

- columns in HTML table
  - determining 5-13
- columns of report 1-14, 1-16
- Columns property 6-97
- combo boxes 6-23
  - changing properties 6-30
  - defining list 6-23 to 6-24
  - placing on forms 6-23
  - setting values dynamically 6-23, 6-25
  - triggering actions 6-23, 6-25
- Comma inclusion edit option 2-23, 2-25
- commands 13-4
  - adding to user-defined toolbars 13-8
  - deleting from toolbars 13-8
  - executing using toolbars 13-4
- Commands tab 13-10
- comments 12-9
  - syntax coloring 12-9
- COMMIT command 11-3
  - DB2 data sources 11-18 to 11-19
  - defining a logical transaction 11-3
- COMMIT parameter 11-10
- communication configuration files 11-16, 11-18
  - for WebFOCUS Servers 11-16, 11-18
- COMPILE command 8-3
- compiling Maintain procedures 8-3
- Computed Stack Column dialog box 2-13, 2-15, 2-17
- concurrent processing 11-9, 11-12
- configuration files for WebFOCUS Servers 11-16, 11-18
  - communication configuration files 11-16, 11-18
  - EDACSG 11-16, 11-18
  - odin.cfg 11-16, 11-18
- confirm function 5-11
- Control Columns dialog box 6-65
- controls 6-1 to 6-2, 6-4
  - adding to forms 3-4
  - aligning 6-7 to 6-10
  - changing tab order 6-111
  - copying 6-4 to 6-5
  - cutting 6-4
  - duplicating 6-4, 6-6
  - grouping 6-10 to 6-11
  - layering 6-12, 6-14
  - manipulating at design time 6-3
  - manipulating at run time 6-106 to 6-107
  - moving between layers 6-12, 6-14
  - pasting 6-4 to 6-5
  - resizing 6-6 to 6-7
  - selecting 6-3
  - sending to the back 6-11 to 6-12
  - sending to the front 6-11
  - spacing 6-10
- Controls palette 3-3 to 3-4, 6-87
  - menu control 6-87
- Cool look option 13-4 to 13-5, 13-9
- copying functions to procedures 2-8
- Create New Stack dialog box 2-17, 2-23
- creating project files 1-4
- creating projects 1-2
- Credit negative edit option 2-23, 2-25
- currency 2-23, 2-25
  - formatting 2-23, 2-25
- Customize dialog box 13-9 to 13-10
- customizing colors in text editor 12-9
- customizing in text editor 12-9

## D

---

- data source descriptions 1-12
  - creating 1-4
  - editing 1-13
  - viewing structure in Project Explorer 1-13 to 1-14
- data source stack columns 6-40
  - adding to read/write grid 6-40
- data source stacks 2-17 to 2-18
  - columns 6-59
  - creating 2-17 to 2-18
  - creating from Select Segment Fields dialog box 6-35, 6-37
  - displaying 6-59
  - displaying implied columns 2-17
  - displaying in a folder 2-1, 2-4
  - editing 2-17, 2-19 to 2-20, 2-22
  - viewing 6-59
- data sources 1-3, 1-12
  - adding to project 1-6
  - command failure 11-4
  - FOCUS Database Server 11-12
  - logical transactions 11-1
  - position within logical transactions 11-5
  - reading with report procedures 11-13
  - removing from project 1-10
  - renaming 1-10
  - sharing 11-9
  - specifying in a procedure 2-4 to 2-5
- data types 2-13, 2-23, 7-2
  - classes 7-2, 7-4 to 7-5, 7-7
- date formats 2-23 to 2-24
- DB2 data sources 11-18
  - change verification 11-18, 11-22, 11-24
  - data adapter differences 11-18 to 11-19
  - transaction processing 11-18, 11-20, 11-22, 11-24
- DEBUG command 8-3 to 8-4
- debug option for deploying Maintain applications 8-3
- debugging WebFOCUS Maintain applications 8-1
- Decimal format 2-23
- Declaration tab in Variable Editor 2-13, 2-15
- DECLARE command 7-15 to 7-16
- deep copy 6-4 to 6-6
- Definition tab 2-17, 2-20
- deleting projects or project components 1-10
- deploying applications 11-16
  - team development 10-2
- DESCRIBE command 7-4, 7-7
- Description tab 2-3, 2-13, 2-17
  - in Class Editor 7-4, 7-13
  - in Function Editor 2-11
  - in Link Editor 5-27
  - in Stack Editor 2-17, 2-22
  - in Variable Editor 2-13, 2-17
- directional searches in text editor 12-6
- directories, adding to project path 1-9
- Display all files in the project paths button 1-9
- docked windows 13-2 to 13-3
- domain name in URL Wizard 5-29
- Double Precision format 2-23, 2-25
- DoubleClick event 5-5
- drag-and-drop operations
  - adding data source stack columns to HTML tables 6-59
  - adding stack columns to read/write grids 6-40
  - copying functions to a procedure 2-8
  - copying text 12-3 to 12-4
  - copying variables to a procedure 2-15
  - creating data source stacks 2-18

drag-and-drop operations (*continued*)

- defining lists in the List Source dialog box 6-23 to 6-24
- duplicating controls 6-4, 6-6
- dynamically setting color properties 6-108 to 6-109
- generating code to display a form 4-5 to 4-6
- generating syntax 12-3, 12-5
- generating WINFORM SET syntax 4-10, 6-106 to 6-107
- inserting IWCTrigger into a script 5-10, 5-12
- moving functions to another procedure 2-6, 2-8
- moving text 12-3, 12-5
- moving variables to another procedure 2-13, 2-15
- placing ActiveX controls on forms 6-100 to 6-101
- placing images on forms 6-71 to 6-72
- placing Java applets on forms 6-81
- setting properties dynamically 3-5, 3-7
- specifying functions as event handlers 5-7 to 5-8
- using an import module in a procedure 2-38 to 2-39

draganddrop operations

- generating WINFORM SET syntax 4-9

drawing grid 3-8

drilling down in HTML Tables 6-68

driver procedures 4-11 to 4-12

## **E**

---

EDACSG ddname 11-16, 11-18

EDAPRINT.LOG file 8-2

EDASPROF global server profile 11-15

- FOCUS Database Server 11-15

edit boxes 6-31

- changing properties 6-34
- placing on forms 6-31

Edit Class dialog box 7-4, 7-10 to 7-12

Edit Function dialog box 2-6, 2-9

Edit menu 6-3, 12-3 to 12-4

Edit parameters command 2-26 to 2-28

Edit Variable dialog box 2-13, 2-15

editing FOCUS procedures 1-15

editing Master Files 1-13

Editor tab 12-7 to 12-8

Editor toolbar 13-4

Editor windows 12-1 to 12-2, 12-6 to 12-7

e-mail address 5-18

Email Address dialog box 5-36

e-mail links 5-18, 5-21, 5-35

e-mail messages 5-37

END keyword in GOTO command 11-6

end-user input 5-10

- using scripts 5-10, 5-13
- validating 5-10

Enter a List Item dialog box 6-23, 6-28

Enter a Parameter dialog box 5-33, 6-81, 6-84

Environment option 12-9 to 12-10, 13-12

environment variables 11-16, 11-18

- HOME 11-16, 11-18

error messages 1-10

Event Handler editor 5-1 to 5-3, 12-1 to 12-2

events 5-1 to 5-2, 5-5, 6-40, 6-43

- common combinations 5-7
- read/write grids and 6-40, 6-43

EXEC command 5-7

- in Event Handler Editor 5-7

## Index

- Explorer 1-7
  - viewing new file types 1-9
- Explorer toolbar
  - Display all files in the project paths 1-9
- External Procedure editor 12-1 to 12-2
- external procedures 1-6, 11-16 to 11-17
  - deployment considerations 11-16
  - executing via IWCLink 9-2
  - reporting 11-16
- F**

---
- FDS (FOCUS Database Server) 11-10
  - change verification 11-12
  - deployment considerations 11-16
  - identifying 11-15
  - report procedures 11-16
  - SET PATHCHECK 11-14
  - transaction processing 11-13
- File menu 12-11
- file types, viewing in Explorer 1-9
- files 1-3
  - adding to project 1-6
  - creating 1-4
  - deleting 1-10
  - removing from project 1-10
  - renaming 1-10
  - viewing in project path 1-9
- Find dialog box 12-10 to 12-11
- Floating dollar edit option 2-23, 2-25
- Floating Point format 2-23, 2-25
- flow of control 4-5
- FocCurrent variable 11-8
  - change-verify protocol 11-12
- FocIndex command 6-68, 6-70
- FOCSET command 8-6
- FOCUS code 1-14
- FOCUS data sources 11-10, 11-15
  - change verification 11-13
  - concurrent transactions 11-10
  - FOCUS Database Server 11-12
  - SET COMMIT 11-10 to 11-11
  - SET PATHCHECK parameter 11-14
  - sharing access 11-12
  - transaction processing 11-10, 11-14
- FOCUS Database Server (FDS) 11-10
  - change verification 11-13
  - deployment considerations 11-16
  - identifying 11-15
  - report procedures 11-16
  - SET PATHCHECK 11-14
  - transaction processing 11-12
- Focus event 5-5
  - in grids 6-43
- FOCUS procedures 1-3
  - adding to project 1-6
  - creating 1-4
  - deleting 1-10
  - editing 1-15
  - removing from project 1-10
  - renaming 1-10
- folders for procedure components 2-1, 2-4
- fonts 6-47, 6-50
  - in read/write grids 6-47, 6-50
  - in text editor 12-10
- Form Editor 3-1 to 3-2
  - applying colors 6-108
  - control handles 6-3
  - grids 3-8
  - layout 3-3
  - selecting all 6-3
- Format data type 2-23

- forms 3-3, 4-1 to 4-2
    - active 4-7
    - changing properties 4-4
    - closing at run time 4-7 to 4-9, 5-2 to 5-3
    - creating 4-2 to 4-3, 10-2
    - creating menus 6-87 to 6-88
    - deleting 4-2
    - displaying at run time 4-5 to 4-7
    - driver procedures 4-11 to 4-12
    - editing 4-2 to 4-3
    - flow of control 4-5
    - manipulating at run time 4-5
    - navigating 4-2
    - non-persistent 4-7
    - persistent 4-7
    - renaming 4-2 to 4-3
    - team development 10-2
    - using script libraries 5-17 to 5-18
  - frames 6-38
    - changing properties 6-39
    - placing on forms 6-38
  - FTP servers 5-18
    - defining a Web link 5-20
    - specifying in URL Wizard 5-29
  - Function Parameter dialog box 2-13, 2-15, 2-17
  - Function Return dialog box 2-13, 2-15, 2-17
  - functions 2-6, 6-47, 7-2
    - changing fonts in read/write grids 6-47, 6-50
    - creating 2-6, 2-11
    - deleting 2-6
    - displaying a click message in read/write grids 6-47, 6-50 to 6-51
    - displaying in a folder 2-1, 2-4
    - dynamically changing read-only access 6-47, 6-53
    - dynamically changing text color in read/write grids 6-47, 6-52
    - editing 2-6 to 2-8
    - focusing cells in read/write grids 6-47, 6-51
    - functions 2-6, 6-47, 7-2 (*continued*)
      - focusing columns in read/write grids 6-47, 6-51
      - focusing rows in read/write grids 6-47, 6-51
      - JavaScript and VBScript 5-8 to 5-10, 5-18
      - member 7-2
      - renaming 2-6
      - restricting access in read/write grids 6-47, 6-50
      - running from Event Handler editor 5-7 to 5-8
      - setting a mask in read/write grids 6-47, 6-52
      - setting text color in read/write grids 6-47, 6-52
      - setting text in read/write grids 6-47, 6-52
      - using in read/write grids 6-47
  - Functions tab in Class Editor 7-4, 7-11
- 
- G**
- General tab
    - Class Editor 7-4
    - Options dialog box 13-12
  - General toolbar 13-4
    - Save 1-8
    - Save all 1-8
    - Search project 1-11
  - GET keyword in WINFORM command 4-9 to 4-10, 6-106
  - GetCurrentColumn function 6-47
  - GetCurrentRow function 6-47
  - GetGridDefault function 6-47
  - gif images 6-71
  - global attributes 6-107
  - GOTO END command 11-6 to 11-7
  - GoToCell function 6-47
  - GoToColumn function 6-47
  - graphics 1-6, 6-71

Grid Column Properties dialog box 6-40

Grid Control Columns dialog box 6-40

grid in Form Editor 3-8  
    changing settings 3-8

Grid Settings dialog box 3-8 to 3-9

grids 6-40, 6-47  
    changing properties 6-46  
    drag-and-drop operations 6-40  
    events 6-40, 6-43  
    functions 6-47  
    placing on forms 6-40

group box controls 6-53  
    changing properties 6-54  
    placing on forms 6-53

group development of applications 10-2

guidelines in Form Editor 3-3, 3-8

## **H**

---

Help dialog box 6-110

Help files 6-110

HIDE keyword in WINFORM command 4-7

HOME environment variable 11-16, 11-18

HTML Content Source dialog box 6-55 to 6-56

HTML File editor 12-1 to 12-2

HTML files 1-3, 5-18  
    adding to project 1-6  
    creating 1-4

HTML Object control 6-55  
    changing properties 6-58  
    displaying FOCUS reports 1-14 to 1-16  
    placing on forms 6-55

HTML Table Columns dialog box 6-59

HTML Table control  
    determining click locations 5-13

HTML tables 6-59, 6-67

    changing properties 6-70  
    creating links 6-68  
    determining cell values 6-68 to 6-69  
    determining click location 6-68 to 6-69  
    determining columns 6-68 to 6-69  
    determining rows 6-68 to 6-69  
    displaying report results 6-59 to 6-60  
    drilling down in 6-68  
    placing on forms 6-59

HTMTABLE format for FOCUS reports 1-14 to 1-16

HTTP Server 5-20

https protocol 5-29

## **I**

---

image control 6-71  
    changing at run time 6-80 to 6-81  
    placing on forms 6-71 to 6-72

Image Map dialog box 6-77, 6-79

image maps 6-77, 6-79

Image Source dialog box 6-71, 6-73

ImageDown property 6-80

ImageOver property 6-80

images 6-71, 6-76

implied columns in a data source stack 2-17 to 2-18

IMPORT keyword in MODULE command 7-14

import modules 1-3, 2-38, 7-14  
    creating 1-4  
    deleting 1-10  
    importing into procedures 2-38 to 2-39, 7-14  
    renaming 1-10  
    saving 1-8  
    using in procedures 2-38 to 2-39

IMS/TM transactions 1-6

inheritance 7-4

Initialize tab in Variable Editor 2-13, 2-16  
 initializing variables 2-13, 2-16  
 input parameters for a procedure 2-26 to 2-29  
 Insert ActiveX Control dialog box 6-100, 6-103  
 Insert ActiveX Control Resource dialog box 6-100, 6-104  
 insert mode in text editor 12-3  
   determining status 13-11  
 Insert Text dialog box 6-31 to 6-32  
 Integer format 2-23, 2-25  
 IWCLink script function 9-2  
 IWCTrigger function 5-10, 5-12  
   passing ActiveX control values 6-100, 6-102

## **J**

---

Java applet controls 6-81  
   changing parameters at run time 6-81, 6-83  
   placing on forms 6-81  
 Java Applet Parameters dialog box 6-81, 6-83  
 Java applets 6-81  
   adding as resources 6-81 to 6-82  
   editing 6-81  
   parameters 6-81, 6-84  
 JavaScript 5-9  
 JavaScript functions 5-8 to 5-9  
   calling 5-9 to 5-10, 5-16  
   debugging 5-18  
   passing ActiveX control values 6-100, 6-102  
   using for validation 5-10  
   using in applications 5-9  
   writing in Event Handler editor 5-9  
 JavaScript libraries 5-17  
   associating with forms 5-17 to 5-18  
   creating 1-4, 5-17  
   editing 5-17 to 5-18  
 jpeg images 6-71

## **K**

---

KEEP keyword 11-6  
 KeyPress event 5-5

## **L**

---

Language Wizard 12-12  
 layers 6-12  
   activating 6-12 to 6-13  
   creating 6-12 to 6-13  
   hiding 6-12 to 6-13  
   locking 6-12, 6-14  
   unhiding 6-12 to 6-13  
   unlocking 6-12, 6-14  
 Layers Sheet dialog box 6-12, 6-14  
 Layout menu 3-3  
 Layout toolbar 13-4  
   Toggle grid 3-8  
   Toggle rulers 3-9  
 Leading zeros edit option 2-23, 2-25  
 libraries 7-14  
   class definitions 7-14  
   importing 7-14  
   Top function 7-14  
 line breaks in HTML Table header titles 6-59, 6-67  
 line of procedure 13-11  
 lines 6-86  
   changing properties 6-87  
   placing on forms 6-86  
 Link Editor dialog box 5-26 to 5-27  
 links 5-18, 5-21  
   email 5-21, 5-35  
   HTML tables 6-68  
   Web 5-22, 5-37

## Index

- list box controls 6-23
  - changing properties 6-30
  - defining list 6-23 to 6-24
  - placing on forms 6-23
  - setting values dynamically 6-23, 6-25
  - triggering actions 6-23, 6-25
- List Source dialog box 6-23, 6-26
- ListItems stack 6-23, 6-25, 6-93, 6-95
- LockColumns function 6-47
- LockRows function 6-47
- logic in applications 9-2
- logical transactions 11-1, 11-9
  - broadcast commit 11-7
  - concurrent transactions 11-9 to 11-10
  - concurrent transactions in FOCUS Database Server 11-12
  - data source position 11-5
  - DBMS types 11-7
  - defining 11-3 to 11-4
  - deployment considerations 11-16
  - ending an application 11-8
  - failure 11-4
  - FocCurrent 11-8 to 11-9
  - multiple data source types 11-7
  - multiple servers 11-8
  - open transactions when an application ends 11-8
  - processing 11-1 to 11-2
  - rolling back 11-5
  - spanning procedures 11-6
  - success 11-7
- M**

---
- Maintain Development Environment 1-1, 1-7
  - customizing 13-1
  - opening 1-2
- Maintain functions 2-6, 2-8
  - editing text 2-6, 2-8
- Maintain language 12-1 to 12-2, 12-12
- Maintain procedures 1-3, 2-1 to 2-2, 11-6, 11-15
  - adding to project 1-6
  - compiling 8-3
  - creating 1-4
  - deleting 1-10
  - editing 2-3
  - removing from project 1-10
  - renaming 1-10
  - saving 1-8
  - selecting starting procedure 2-2
  - setting parameters 2-26 to 2-29
  - specifying data sources 2-4 to 2-5
  - transaction integrity 11-6
  - using import modules 2-38 to 2-39
  - viewing components in folders 2-4
- Maintain statement trace 8-2
- Map property 6-77 to 6-78
- Mask function 6-47, 6-49
- masks 6-47, 6-49, 6-52
- Master Files 1-3, 1-12
  - adding to project 1-6
  - creating 1-4
  - deleting 1-10
  - editing 1-13
  - removing from project 1-10
  - renaming 1-10
  - viewing structure in Project Explorer 1-13
- Member Function dialog box 2-6, 2-9, 2-11
- member functions 7-2
  - inheritance 7-4
- Member Variable dialog box 2-13, 2-15, 2-17
- member variables 7-2 to 7-3
  - inheritance 7-4

- menu bars 6-87 to 6-88
  - adding 6-87 to 6-88
  - changing properties 6-92
  - submenus 6-87, 6-89

Menu control 6-87

Menu Items dialog box 6-87, 6-91

menus

- adding 6-88
- changing properties 6-92
- submenus 6-87

Miscellaneous toolbar 13-4

- Tip of the day 13-15

MNTPERF setting 8-2

MNTSTMT setting 8-2 to 8-6

MNTSTMT.MAS file 8-6

MODIFY applications 11-18

MODULE command 7-14

monetary values 2-23, 2-25

MouseDown event 5-5

MouseMove event 5-5

MouseOut event 5-5

MouseOver event 5-5

MouseUp event 5-5

moving functions to another procedure 2-6, 2-8

multi-edit boxes 6-31

- changing properties 6-34
- placing on forms 6-31

## **N**

---

Name property 4-2 to 4-3

New Class dialog box 7-4, 7-10

New Function dialog box 2-9, 2-11

New project file dialog box 1-5

New Toolbar dialog box 13-8

New Variable dialog box 2-13, 2-15 to 2-17

New Window Options dialog box 5-2, 5-25

Next Bookmark icon 12-10 to 12-11

Num Lock 13-11

numeric formats 2-23, 2-25

## **O**

---

objects 7-1, 7-15

- declaring 7-15 to 7-16
- specifying in URL Wizard 5-31
- Variable Editor 7-15

odin.cfg file 11-16, 11-18

OnCanViewMove event 6-43

OnCBDClicked event 6-43

OnCBLClicked event 6-43

OnCBRClicked event 6-43

OnCellChange event 6-43

OnCellChanged event 6-43

OnCharDown event 6-43

OnColChange event 6-43

OnColSized event 6-43

OnEditFinish event 6-43

OnEditStart event 6-43

OnHitButtom event 6-43

OnKeyDown event 6-43

OnLClicked event 6-43

OnRClicked event 6-43

OnRowChange event 6-43

OnRowSize event 6-43

## Index

- OnRowSizing event 6-43
  - OnSHDClicked event 6-43
  - OnSHLClicked event 6-43
  - OnSHRClicked event 6-43
  - OnTHDClicked event 6-43
  - OnTHLClicked event 6-43
  - OnTHRClicked event 6-43
  - Open dialog box 1-22
  - Open event 5-5
  - Options dialog box 13-12
    - Editor tab 12-7 to 12-8
    - General tab 13-12
  - Oracle procedures 1-6
  - output parameters 2-26 to 2-28
  - Output window 1-10 to 1-11
  - overtyping mode in text editor 12-3, 13-11
- P**
- 
- panes 13-2
    - deleting from windows 13-2 to 13-3
    - splitting windows into 13-2 to 13-3
  - parameters 2-26
    - adding 5-20
    - binding 5-20
    - setting 2-26, 2-28 to 2-29
    - specifying in URL Wizard 5-32
  - Parameters dialog box 6-81, 6-83
  - parent target names 5-2 to 5-3
  - Paste Appearance command 6-4 to 6-5
  - PATHCHECK parameter 11-14
  - paths in project
    - changing 1-9
    - specifying at creation time 1-2
  - PCHOLD files 1-14 to 1-15
  - performance transaction processing 11-5
  - pictures 6-71
  - Previous Bookmark command 12-10 to 12-11
  - Procedure Editor 12-1 to 12-2
  - procedures, Maintain 1-3, 2-1 to 2-2
    - adding to project 1-6
    - creating 1-4
    - deleting 1-10
    - editing 2-3
    - MODIFY 11-18
    - removing from project 1-10
    - renaming 1-10
    - saving 1-8
    - selecting starting procedure 2-2
    - setting parameters 2-26 to 2-29
    - sharing data sources with WebFOCUS Maintain 11-18
    - specifying data sources 2-4 to 2-5
    - using import modules 2-38 to 2-39
    - viewing components in folders 2-4
  - procedures, WebFOCUS 1-3
    - adding to project 1-6
    - creating 1-4
    - deleting 1-10
    - editing 1-15
    - removing from project 1-10
    - renaming 1-10
  - project components 1-3
    - adding to project 1-6
    - creating 1-4
    - deleting 1-10
    - removing 1-10
    - renaming 1-10
    - saving 1-8
  - Project Explorer 1-7
    - displaying data source description structure 1-13 to 1-14

Project Explorer tab 13-12, 13-14

projects 1-2

changing paths 1-2, 1-9

components 1-3

creating 1-2

defining flow 1-7

deleting 1-10

removing components 1-10

renaming 1-10

saving 1-8

searching text 1-11

Starting Object 2-2

viewing/hiding files in path 1-9

prompt function 5-10, 5-12

prompted edit boxes 6-35

properties 3-5, 4-9 to 4-10, 6-106

changing 3-7

changing dynamically 3-7

determining at run time 4-9 to 4-10, 6-106

setting at run time 4-9 to 4-10, 6-106 to 6-107

property sheet 3-3, 3-5

ActiveX controls 6-100

pull-down menus 6-87 to 6-89

## Q

---

QuickSetAlignment function 6-47

QuickSetBackColor function 6-47

QuickSetFont function 6-47

QuickSetTextColor function 6-47

QuickSetTextColor property 6-47

## R

---

radio button controls 6-93

changing properties 6-96

determining layout 6-97

placing on forms 6-93 to 6-94

radio button controls 6-93 (*continued*)

setting values dynamically 6-93, 6-95

triggering actions 6-93, 6-95

RDBMS stored procedures 1-6, 1-14

read/write grids 6-40 to 6-41, 6-47

adding stack columns 6-40

changing properties 6-46

events 6-40, 6-43

functions 6-47

placing on forms 6-40 to 6-41

Redo command 6-12, 12-3, 12-5

RedrawAll function 6-47

REFRESH keyword in WINFORM command 4-9

report columns 1-14, 1-16

reports 1-14, 5-20

defining Web link 5-20

reading transaction data 11-13

specifying in URL Wizard 5-30

RESET keyword 4-9, 4-11

WINFORM command and 4-9, 4-11

Resource Wizard 5-19

resources

ActiveX controls 6-100 to 6-101, 6-105

HTML files 5-18

images 6-71, 6-73

Java applets 6-81 to 6-82

return values for functions 2-6

return values for procedures 2-26 to 2-29

ROLLBACK command 11-3, 11-5

DB2 data sources 11-18 to 11-19

rows in HTML table, determining 5-13

Rows property 6-97

rulers 3-3, 3-8 to 3-9

## S

---

- Save all button/command 1-8
- Save button/command 1-8
- saving projects 1-8
- scalable applications 9-2
- Scientific notation edit option 2-23, 2-25
- scope of variables 2-6
- Script editor 12-1 to 12-2
- script functions 5-8 to 5-9
  - debugging 5-18
  - for validation 5-10
  - in applications 5-9
  - running 5-9 to 5-10, 5-16
  - writing in Event Handler editor 5-9
- script libraries 5-17
  - associating with forms 5-17 to 5-18
  - creating 5-17
  - editing 5-17 to 5-18
- scripts 5-8
- Search project command 1-11
- Search project dialog box 1-11 to 1-12
- searching for text 12-6
- select control 3-4
- Select Segment Fields dialog box 6-35, 6-37
- Select Stack Columns dialog box 6-35 to 6-36
- SELF 5-2 to 5-3
- self target name 5-2 to 5-3
- self.WinClose function 4-7 to 4-8, 5-2 to 5-3
- self.WinExit function 4-7 to 4-8, 4-11, 5-2 to 5-3
- servers
  - configuring server instances 9-2
  - Shared Application Servers 9-2
  - tracing 8-5
- Set Check Box State dialog box 6-18, 6-20
- SET keyword in WINFORM command 4-9 to 4-10, 6-106 to 6-107
- SET parameters 11-10
  - COMMIT in FOCUS data sources 11-10 to 11-11
  - PATHCHECK 11-14
  - TRACEOFF 8-5
  - TRACEON 8-2, 8-5 to 8-6
  - TRACEUSER 8-5 to 8-6
- SET PATHCHECK parameter 11-14
- SetCurrentColumn function 6-47
- SetCurrentColumn property 6-47
- SetCurrentRow function 6-47
- SetCurrentRow property 6-47
- SetTHNumberRows function 6-47
- SetTHNumberRows property 6-47
- SetTHRowHeight function 6-47
- SetTHRowHeight property 6-47
- shallow copy 6-4 to 6-5
- Shared Application Servers 9-1 to 9-2, 9-4
  - deploying applications 9-2
  - designing transactions 9-2 to 9-3
  - logic guidelines 9-2
- shortcuts
  - adding Web links to Form Editor 5-22
  - copying functions to a procedure 2-8
  - copying text 12-3 to 12-4
  - copying variables to a procedure 2-13, 2-15
  - creating data source stacks 2-17 to 2-18
  - defining lists in the List Source dialog box 6-23 to 6-24
  - duplicating controls 6-4, 6-6
  - dynamically setting color properties 6-108 to 6-109
  - generating code to display a form 4-5 to 4-6

shortcuts (*continued*)

- generating WINFORM SET syntax 4-9 to 4-10, 6-106 to 6-107
- inserting IWCTrigger into a script 5-10, 5-12
- moving functions to another procedure 2-6, 2-8
- moving text 12-3, 12-5
- moving variables to another procedure 2-13, 2-15
- placing ActiveX controls on forms 6-100 to 6-101
- placing images on forms 6-71 to 6-72
- placing Java applets on forms 6-81
- setting properties dynamically 3-5, 3-7
- specifying a function as an event handler 5-7 to 5-8
- using import modules in procedures 2-38 to 2-39

Show All Files command 1-6, 1-9

Show description 5-23

SHOW keyword in WINFORM command 4-5 to 4-6

Show Tooltips option 13-4 to 13-5, 13-9

SHOW\_ACTIVE keyword in WINFORM command 4-5 to 4-6

SHOW\_AND\_EXIT keyword 4-5 to 4-6

SHOW\_INACTIVE keyword 4-5, 4-7, 6-106 to 6-107

- color settings and 6-108 to 6-109

- WINFORM SET and 6-106 to 6-107

Signature tab in Function Editor 2-6, 2-9

signatures for functions 2-6

spacing controls 6-10

Split command 13-2 to 13-3

Stack Editor 2-17, 2-20, 2-22

Stack Overflow error 4-11

## stacks 2-17

- creating 2-17 to 2-18

- displaying in a folder 2-1, 2-4

- editing 2-17, 2-19

- viewing 6-59

Starting Object for project 2-2

starting procedure 2-1 to 2-2

Status bar 13-11

stored procedures 1-14

submenus 6-87 to 6-89

Sybase procedures 1-6

syntax color 12-9

- turning on or off 12-9

SYS\_MGR.FOCSET command 8-6

system messages 1-10

**T**


---

Tab Order 6-111

Tab Order dialog box 6-111 to 6-112

Table Column dialog box 6-59, 6-67

tabs in text editor 12-10

team development 10-1 to 10-2

text 12-1 to 12-2

- editing 12-1 to 12-2

text controls 6-98

- changing properties 6-99

- placing on forms 6-98

- using as hyperlinks 6-100

text editor windows 12-7

text editors 12-1 to 12-2, 12-6

- changing colors 12-9

- changing fonts 12-10

- copying 12-3 to 12-4

- cutting 12-4

## Index

text editors 12-1 to 12-2, 12-6 (*continued*)  
    deleting text 12-5  
    inserting 12-3  
    matching case searches 12-6  
    pasting text in 12-3 to 12-4  
    printing 12-11 to 12-12  
    replacing text 12-6 to 12-7  
    setting tabs 12-10  
    viewing 12-11

text files 1-3  
    creating 1-4

text searches 1-11

Tile horizontally command 13-2 to 13-3

Tile vertically command 13-2 to 13-3

Tip of the day 13-15

Toggle Bookmark command 12-10 to 12-11

Toggle grid button 3-8

Toggle rulers button 3-9

toolbars 13-4 to 13-5  
    adding commands 13-8  
    built-in 13-4  
    creating 13-7 to 13-8  
    customizing 13-9 to 13-10  
    deleting commands 13-8  
    displaying 13-4 to 13-5  
    executing commands 13-4  
    hiding 13-4 to 13-5, 13-7  
    in moveable windows 13-4 to 13-5  
    renaming 13-7

Toolbars tab 13-9

Tooltips option 13-4 to 13-5, 13-9

Top function 7-14  
    libraries and 7-14

TRACEON setting 8-6

TRACEUSER setting 8-6

tracing 8-2, 8-5, 8-7  
    setting server traces from the Web Console 8-5  
    viewing output 8-6

transaction integrity 11-1, 11-3, 11-9, 11-18  
    across procedures 11-6  
    change verification 11-13  
    concurrent transactions 11-9  
    deployment considerations 11-16  
    FocCurrent 11-8  
    multiple servers 11-8

transaction locking strategies 11-18, 11-20, 11-22

transaction processing 11-1, 11-9  
    broadcast commit 11-7  
    change verification 11-13  
    concurrent transactions 11-12  
    multiple data source types 11-7  
    performance 11-5  
    USE command 11-15

transactions 11-6  
    open 11-6  
    spanning procedures 11-6

trc file 8-6

TYPE ON EDAPRINT command 8-2

Type Wizard 2-23

## U

---

UnCheck event 5-5

Undo command 6-12, 12-3, 12-5

undocked windows 13-2 to 13-3

UNHIDE keyword in WINFORM command 4-5

Universal Resource Locators (URLs) 5-26

URL Link dialog box 6-39

URL tab in Link Editor 5-26

URL Wizard 5-19 to 5-20, 5-28, 5-31  
 entering an email message 5-37  
 including email links 5-35  
 specifying amper variables 5-32  
 specifying parameters 5-32

URLs (Universal Resource Locators) 5-26

USE command with FOCUS Database Server 11-15

user-defined toolbars 13-7 to 13-8

## V

---

variable-length alphanumeric format  
 using with multi-edit boxes 6-31

variables 2-13, 7-2, 7-15  
 copying to a procedure 2-13, 2-15  
 creating 2-13 to 2-14  
 defining objects 7-15 to 7-16  
 defining objects 7-15  
 deleting 2-13  
 displaying in a folder 2-1, 2-4  
 editing 2-13 to 2-14  
 initializing 2-13, 2-16  
 moving to another procedure 2-13, 2-15  
 renaming 2-13  
 viewing data type in Application Explorer  
 2-13

Variables tab in Class Editor 7-4, 7-12

variablelength alphanumeric format 2-23 to 2-24

VB Scripts  
 creating 1-4

VBScript functions 5-8 to 5-9  
 debugging 5-18  
 passing ActiveX control values 6-100, 6-102  
 running 5-9 to 5-10  
 using for validation 5-10  
 using in applications 5-9  
 writing in Event Handler editor 5-9

VBScript libraries 5-17  
 associating with forms 5-17 to 5-18

VBScript libraries 5-17 (*continued*)  
 creating 5-17  
 editing 5-17 to 5-18

View menu 13-4  
 status bar 13-11  
 toolbars 13-4 to 13-5

View Toolbars dialog box 13-4 to 13-5

viewing trace output 8-6

## W

---

Web links 5-18, 5-37  
 adding to a form 5-22  
 creating 5-19  
 deleting 5-23  
 editing 5-22  
 using as event handlers 5-22  
 viewing in browser 5-22

WebFOCUS Developer Studio 1-2

WebFOCUS Maintain 1-1  
 opening 1-2

WebFOCUS procedures  
 adding to project 1-6  
 creating 1-4  
 deleting 1-10  
 editing 1-15  
 removing from project 1-10  
 renaming 1-10

WebFOCUS reports 5-20  
 defining Web link 5-20, 6-81, 6-84  
 parameters 5-32, 6-81, 6-84  
 specifying in URL Wizard 5-30

WebFOCUS Servers  
 communication configuration files 11-16,  
 11-18  
 configuration directory 11-16, 11-18  
 EDACSG 11-16, 11-18  
 edaserve 11-16, 11-18  
 FOCUS Database Server profiles 11-15

## Index

WebFOCUS Servers (*continued*)  
    logical transactions spanning servers 11-8  
    odin.cfg 11-16, 11-18  
    specifying in URL Wizard 5-29

wildcard searches in text editor 12-6

WinClose function 4-7 to 4-8, 5-2 to 5-3

windows 13-2  
    active 13-2  
    cascading 13-2, 13-4  
    deleting panes 13-2 to 13-3  
    docking 13-2  
    splitting 13-2 to 13-3  
    tiling 13-2 to 13-3  
    undocking 13-2 to 13-3

Windows clipboard 6-4 to 6-5, 12-3 to 12-4

WinExit function 4-11, 5-2 to 5-3

WINFORM command 4-5  
    CLOSE 4-7 to 4-8  
    CLOSE\_ALL 4-7, 4-9  
    flow of control 4-5  
    GET 4-9 to 4-10, 6-106  
    HIDE 4-7  
    REFRESH 4-9  
    RESET 4-9, 4-11  
    SET 4-9 to 4-10, 6-106 to 6-107  
    SHOW\_AND\_EXIT 4-5 to 4-6  
    SHOW\_INACTIVE 4-5, 4-7  
    UNHIDE 4-5

wizards  
    Language Wizard 12-12  
    Resource Wizard 5-19  
    Type Wizard 2-23  
    URL Wizard 5-19, 5-28

wrap around searches in text editor 12-6

## **X**

---

x=? initialize tab in Variable Editor 2-13, 2-16

## **Z**

---

Zero suppression edit option 2-23, 2-25

---

---

## Reader Comments

In an ongoing effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. You can contact us through the following methods:

- Mail:** Documentation Services - Customer Support  
Information Builders, Inc.  
Two Penn Plaza  
New York, NY 10121-2898
- Fax:** (212) 967-0460
- E-mail:** [books\\_info@ibi.com](mailto:books_info@ibi.com)
- Web form:** <http://www.informationbuilders.com/bookstore/derf.html>

Name: \_\_\_\_\_

Company: \_\_\_\_\_

Address: \_\_\_\_\_

Telephone: \_\_\_\_\_ Date: \_\_\_\_\_

E-mail: \_\_\_\_\_

Comments:

---

---

## Reader Comments