Honeywell

Experion PKS
GUS Basic Script User's Guide

EPDOC-XX41-en-431A
February 2015

Release 431

Honeywell

Document	Release	Issue	Date
EPDOC-XX41-en-431A	431	0	February 2015

Disclaimer

Contents

1 About This Document

This document describes how to,

- Record, edit, and debug scripts.
- Edit custom dialog boxes.

2 Editing and Debugging Scripts

This chapter explains how to use Script Editor, a tool that enables you to edit and debug your BasicScript scripts. It begins with some general information about working with Script Editor and then discusses editing your scripts, running your scripts to make sure they work properly, debugging them if necessary, and exiting from Script Editor.

Contents

Related topics

2.1 Script Editor Basics

This section provides general information that will help you work most effectively with Script Editor. It includes an overview of Script Editor's application window—the interface you'll use to edit, run, and debug your BasicScript scripts—as well as lists of keyboard shortcuts and information on using the Help system.

2.1.1 Script Editor's Application Window

Script Editor's application window contains the following elements:

- Toolbar: a collection of tools that you can use to provide instructions to Script Editor, as discussed in the following subsection.
- Edit pane: a window containing the BasicScript code for the script you are currently editing.
- Watch pane: a window that opens to display the watch variable list after you have added one or more variables to that list.
- Pane separator: a divider that appears between the edit pane and the watch pane when the watch pane is open.
- Status bar: displays the current location of the insertion point within your script.

2.1.2 Toolbar

The following list briefly explains the purpose of each of the tools on Script Editor's toolbar. These tools are discussed in more detail later in the chapter, in the context of the procedures in which they are used.

Tool		Function
	Cut	Removes the selected text from the script and places it on the Clipboard.
	Copy	Copies the selected text, without removing it from the script, and places it on the Clipboard
	Paste	Inserts the contents of the Clipboard at the current position of the insertion point
	Undo	Reverses the effect of the preceding editing change(s)
	Start	Begins execution of a script.
	Break	Suspends execution of an executing script and places the instruction pointer on the next line to be executed.
	End	Stops execution of a script.
	Toggle Breakpoint	Adds or removes a breakpoint on a line of BasicScript code.

Tool		Function
	Add Watch	Displays the Add Watch dialog box, in which you can specify the name of a BasicScript variable. That variable, together with its value (if any), is then displayed in the watch pane of Script Editor's application window.
	Calls	Displays the list of procedures called by the currently executing BasicScript script. Available only during break mode.
	Single Step	Executes the next line of a BasicScript script and then suspends execution of the script. If the script calls another BasicScript procedure, execution will continue into each line of the called procedure.
	Procedure Step	Executes the next line of a BasicScript script and then suspends execution of the script. If the script calls another BasicScript procedure, BasicScript will run the called procedure in its entirety.

Tip
If you forget the name of a toolbar tool, pass the pointer over the tool to display its name

2.1.3 Keyboard Shortcuts

The following lists present various types of keyboard shortcuts, including general shortcuts, navigating shortcuts, editing shortcuts, and debugging shortcuts.

2.1.4 General Shortcuts

Key(s)	Function
F1	Provides context-sensitive help for selected menu commands and variables in the watch pane, for BasicScript terms in the edit pane that have been selected or that contain the insertion point, and for displayed dialog boxes.
Shift+F1	Toggles the Help pointer.
Ctrl+F	Displays the Find dialog box, which allows you to specify text for which you want to search.
F3	Searches for the next occurrence of previously specified text. If you have not previously specified text for which you want to search, displays the Find dialog box.
Ctrl+G	Invokes the Goto Line dialog box.
Esc	Deactivates the Help pointer if it is active. Otherwise, compiles your script and returns you to the host application.

Navigating Shortcuts

Key(s)	Function
Up arrow	Moves the insertion point up one line.
Down arrow	Moves the insertion point down one line.
Left arrow	Moves the insertion point right by one character position.
Right arrow	Moves the insertion point left by one character position.
PgUp	Moves the insertion point up by one window.
PgDn	Moves the insertion point down by one window.
Ctrl+PgUp	Scrolls the insertion point left by one window.

Key(s)	Function
Ctrl+PgDn	Scrolls the insertion point right by one window.
Ctrl + Left arrow	Moves the insertion point to the start of the next word to the left.
Ctrl + Right arrow	Moves the insertion point to the start of the next word to the right.
Home	Places the insertion point before the first character in the line.
End	Places the insertion point after the last character in the line.
Ctrl+Home	Places the insertion point before the first character in the script.
End	Places the insertion point after the last character in the line.
Ctrl+Home	Places the insertion point before the first character in the script.
Ctrl+End	Places the insertion point after the last character in the script.

Editing Shortcuts

Key(s)	Function
Del	Removes the selected text or removes the character following the insertion point without placing it on the Clipboard.
BkSp	Removes the selected text or removes the character preceding the insertion point without placing it on the Clipboard.
Ctrl+A	Selects all text in the script.
Ctrl+Y	Deletes the entire line containing the insertion point without placing it on the Clipboard.
Tab	Inserts a tab character.
Enter	Inserts a new line, breaking the current line.
Ctrl+C	Copies the selected text, without removing it from the script, and places it on the Clipboard.
Ctrl+X	Removes the selected text from the script and places it on the Clipboard.
Ctrl+V	Inserts the contents of the Clipboard at the location of the insertion point.
Shift + any navigating shortcut	Selects the text between the initial location of the insertion point and the point to which the keyboard shortcut would normally move the insertion point. (For example, pressing Shift + Ctrl + Left arrow selects the word to the left of the insertion point; pressing Shift+Ctrl+Home selects all the text from the location of the insertion point to the start of your script.)
Ctrl+Z	Reverses the effect of the preceding editing change(s).

Debugging Shortcuts

Key(s)	Function
Shift+F9	Displays the Add Watch dialog box, in which you can specify the name of a BasicScript variable. Script Editor then displays the value of that variable, if any, in the watch pane of its application window.
Del	Removes the selected watch variable from the watch pane.

Key(s)	Function
Enter or F2	Displays the Modify Variable dialog box for the selected watch variable, which enables you to modify the value of that variable.
F5	Runs your script.
F6	If the watch pane is open, switches the insertion point between the watch pane and the edit pane.
F8	Activates the Single Step command, which executes the next line of a BasicScript script and then suspends execution of the script. If the script calls another BasicScript procedure, execution will continue into each line of the called procedure.
Shift+F8	Activates the Procedure Step command, which executes the next line of a BasicScript script and then suspends execution of the script. If the script calls another BasicScript procedure, BasicScript will run the called procedure in its entirety.
Ctrl+Break	Suspends execution of an executing script and places the instruction pointer on the next line to be executed.
F9	Sets or removes a breakpoint on the line containing the insertion point.

2.2 Using the Help System in Script Editor

Script Editor's Help system provides context-sensitive help both on BasicScript keywords and on how to use various features of Script Editor. The Help system also lets you pinpoint information on key topics. This subsection describes several ways to get help.

To get context-sensitive help using the Help pointer

Here's how to activate the Help pointer and use it to get help on certain key features of Script Editor, including the status bar, toolbar, menu commands, edit pane, watch pane, and pane separator.

1. When you pass the pointer over an area of Script Editor's application window on which you can obtain help, a question mark appears beside the mouse pointer to indicate that the Help pointer is active. To activate the Help pointer, press Shift+F1.

Attention
You can use the Help pointer to obtain help on **all**toolbar tools and menu commands, including those that are currently disabled.

2. Place the Help pointer on the item for which you want help and click the mouse button. Script Editor's Help system displays information for the item on which you clicked.

To get context-sensitive help using the keyboard

Here's how to use the keyboard to get context-sensitive help on BasicScript terms, watch variables, menu commands, and dialog boxes

1. Select the BasicScript term on which you want help or place the insertion point anywhere in the term Or, Select the watch variable or menu command on which you want help (including commands that are currently disabled), Or Display the dialog box on which you want help.
2. Press F1.

   Script Editor's Help system displays information on the BasicScript term, watch variable, menu command, or dialog box.

To search for help on a specific topic

Here's how to access the Help system and pinpoint specific information within it.

1. From the Help menu, choose the Search for Help on command.

   The Search dialog box appears.
2. Either enter the desired topic or select it from the following scrollable list.
3. Click the Show Topics button or press Enter.

   The topic you selected is displayed in the second scrollable list, together with closely related Help topics, if any.
4. Click the Go To button or press Enter.

   Help is displayed for the topic you selected.

To use the Help system contents

Here's how to display the Help system contents and use it to find information on selected topics.

1. From the Help menu, choose Contents.

   A list of major topics in Script Editor's Help system appears.
2. Select the topic on which you want help.

The Help system either displays information on the selected topic or presents a list of more specific subtopics from which you can choose to obtain the desired information.

2.3 Editing Your Scripts

This section explains how to use Script Editor to edit BasicScript code. Although, in some respects, editing code with Script Editor is like editing regular text with a word-processing program, Script Editor also has certain capabilities specifically designed to help you edit BasicScript code.

You'll learn how to move around within your script, select and edit text, add comments to your script, break long BasicScript statements across multiple lines, search for and replace selected text, and perform a syntax check of your script. The section ends with a brief discussion of editing dialog box templates, which is explained in much more detail in Chapter 3.

2.3.1 Navigating within a Script

The lists of keyboard shortcuts in the preceding section contain a group of navigating shortcuts, which you can use to move the insertion point around within your script. When you move the insertion point with a keyboard shortcut, Script Editor scrolls the new location of the insertion point into view if it is not already displayed.

You can also reposition the insertion point with the mouse and the Goto Line command, as explained below.

Script Editor differs from most word-processing programs in that it allows you to place the insertion point anywhere within your script, including in “empty spaces.” (Empty spaces are areas within the script that do not contain text, such as a tab's expanded space or the area beyond the last character on a line.)

To move the insertion point with the mouse

1. Use the scroll bars at the right and bottom of the display to scroll the target area of the script into view if it is not already visible.
2. Place the mouse pointer where you want to position the insertion point.
3. Click the left mouse button.

   The insertion point is repositioned.

   This approach is especially fast if the area of the screen to which you want to move the insertion point is currently visible.

> **Attention**
> When you scroll the display with the mouse, the insertion point remains in its original position until you reposition it with a mouse click. If you attempt to perform an editing operation when the insertion point is not in view, Script Editor automatically scrolls the insertion point into view before performing the operation.

To move the insertion point to a specified line in your script

1. Press F4.

   Script Editor displays the Goto Line dialog box.

2. Enter the number of the line in your script to which you want to move the insertion point.
3. Click the OK button or press Enter.

   This approach is especially fast if the area of the screen to which you want to move the insertion point is not currently visible but you know the number of the target line.

   The insertion point is positioned at the start of the line you specified. If that line was not already displayed, Script Editor scrolls it into view.

> **Attention**
> The insertion point cannot be moved so far below the end of a script as to scroll the script entirely off the display. When the last line of your script becomes the first line on your screen, the script will stop scrolling, and you will be unable to move the insertion point below the bottom of that screen.

2.4 Performing Editing Operations with Script Editor

This subsection provides an overview of the editing operations you can perform with Script Editor—including inserting, selecting, deleting, cutting, copying, and pasting material—and explains how to undo, or reverse, the most recent editing operations. You may wish to refer to the lists of keyboard shortcuts in the preceding section, which contain a group of editing shortcuts that can be used to perform many of the operations discussed here.

2.4.1 Inserting Text

In Script Editor, inserting text and other characters such as tabs and line breaks works about the same way as it does in a word-processing program: you position the insertion point at the desired location in the script and start typing.

However, as noted in the preceding subsection, Script Editor lets you position the insertion point in "empty spaces," which means that you can also insert text into empty spaces—a feature that comes in handy when you want to insert a comment in the space beyond the end of a line in your script. (Adding comments to your script is discussed later in this section.) When you insert characters beyond the end of a line, the space between the insertion point and the last character on the line is backfilled with tab characters.

Another way in which Script Editor differs from word-processing programs is that in Script Editor, text does not wrap. If you keep entering text on a given line, eventually you will reach a point at which you can enter no more text on that line. Therefore, you control the line breaks by pressing Enter when you want to insert a new line in your script. The effect of pressing Enter depends on where the insertion point is located at the time:

- If you press Enter with the insertion point at or beyond the end of a line, a new line is inserted after the current line.
- If you press Enter with the insertion point at the start of a line, a new line is inserted before the current line.
- If you press Enter with the insertion point within a line, the current line is broken into two lines at that location.

If you press Tab, a tab character is inserted at the location of the insertion point, which causes text after the tab to be moved to the next tab position. If you insert new text within a tab's expanded space, the text that originally appeared on that line is moved to the next tab position each time the new text that you are entering reaches the start of another tab position.

When you enter certain types of text in Script Editor, the text automatically appears in a distinctive color. The default colors, which you can change, are as follows:

- When you type keywords, they appear in blue.
- When you type identifier text, it appears in black.
- When you type comments (beginning with either an apostrophe or "REM"), they appear in green.

2.4.2 Selecting Text

You can use either the mouse or the keyboard to select text and other characters in your script. Regardless of which method you use, you should be aware that in Script Editor, you can select either a portion of one line or a series of whole lines, but you cannot select a portion of one line plus one or more whole lines. When you are selecting multiple lines and start or end your selection partway through a line, Script Editor automatically extends the selection to include the entire starting and ending lines.

To select text with the mouse

1. Place the mouse pointer where you want your selection to begin.
2. While pressing the left mouse button, drag the mouse until you reach the end of your selection, and release the mouse button or

   While pressing Shift, place the mouse pointer where you want your selection to end and click the left mouse button.

The selected text is highlighted on your display.

Another way to select one or more whole lines with the mouse is to start by placing the mouse pointer in the left margin beside the first line you want to select. The pointer becomes a reverse arrow, which points toward the line of text. Click the left mouse button to select a single line; press the left mouse button and drag up or down to select multiple lines.

To select text with the keyboard

1. Here's how to use keyboard shortcuts to select text in your script.
2. Place the insertion point where you want your selection to begin.
3. While pressing Shift, use one of the navigating keyboard shortcuts to extend the selection to the desired ending point.
   The selected text is highlighted on your display.

> **Attention**
> When you intend to select an entire single line of text in your script, it is important to remember to extend your selection far enough to include the hidden end-of-line character, which is the character that inserts a new line in your script.

To select an entire line of text with the keyboard

1. Place the insertion point at the beginning of the line you want to select.
2. Press Shift + Down arrow.
   The entire line, including the end-of-line character, is selected.
3. To extend your selection to include additional whole lines of text, repeat step 2.
   Once you have selected text within your script, you can perform a variety of other editing operations on it, including deleting the text, placing it on the Clipboard (either by cutting the text or copying it), and pasting it.

2.4.3 Deleting Text

When you delete material, it is removed from your script without being placed on the Clipboard.

To delete text

- Here's how to remove one or more characters, selected text, or entire lines from your script.
  - To remove a single character to the left of the insertion point, press BkSp once; to remove a single character to the right of the insertion point, press Del once. To remove multiple characters, hold down BkSp or Del.
  - To remove text that you have selected, press BkSp or Del.
  - To remove an entire line, place the insertion point in that line and press Ctrl+Y.

To combine the current line with the following line

1. Place the insertion point after the last character on the current line.
2. Press Del once to delete the hidden end-of-line character.
   The current line and the following line are combined.

Attention

If any spaces were entered at the end of the current line, you may have to press Del one or more additional times to remove these hidden characters first before you can delete the end-of-line character.

Pressing BkSp with the insertion point at the start of a line has no effect—that is, it will not combine the current line with the preceding line.

2.4.4 Cutting and Copying Text

You can place material from your script on the Clipboard by either cutting it or copying it.

Here's how to place on the Clipboard text that you have cut from your script.

To cut a selection

- Press Ctrl+X.
  The selection is removed from your script and placed on the Clipboard.

To copy a selection and place on the Clipboard

- Press Ctrl+C.
  The selection remains in your script, and a copy of it is placed on the Clipboard.

2.4.5 Pasting Text

Once you have cut or copied material to the Clipboard, here's how to paste it into your script at another location.

To paste the contents of the Clipboard into your script

1. Position the insertion point where you want to place the contents of the Clipboard.
2. Press Ctrl+V.
   The contents of the Clipboard appear at the location of the insertion point.

   If you wish to delete a block of text and insert the contents of the Clipboard in its place, you can combine the two operations by first selecting the text you want to remove and then pressing Ctrl+V to replace it with the contents of the Clipboard.

2.5 Undoing Editing Operations

You can undo editing operations that produce a change in your dialog box, including:

- The addition of a control
- The insertion of one or more controls from the Clipboard
- The deletion of a control
- Changes made to a control or dialog box, either with the mouse or with the Information dialog box

You cannot undo operations that don't produce any change in your dialog box, such as selecting controls or dialog boxes and copying material to the Clipboard.

To undo an editing operation:

- Press Ctrl+Z.
  Your dialog box is restored to the way it was before you performed the editing operation.

2.6 Adding Comments to Your Script

You can add comments to your script to remind yourself or others of how your code works. Comments are ignored when your script is executed.

In BasicScript, the apostrophe symbol (') is used to indicate that the text from the apostrophe to the end of the line is a comment.

To add a full-line comment

1. Type an apostrophe (') at the start of the line.
2. Type your comment following the apostrophe.

   When your script is run, the presence of the apostrophe at the start of the line will cause the entire line to be ignored.

   Here's how to designate the last part of a line as a comment.

To add a comment at the end of a line of code

1. Position the insertion point in the empty space beyond the end of the line of code.
2. Type an apostrophe (').
3. Type your comment following the apostrophe.

Attention

When your script is run, the code on the first portion of the line will be executed, but the presence of the apostrophe at the start of the comment will cause the remainder of the line to be ignored.

Although you can place a comment at the end of a line containing executable code, you cannot place executable code at the end of a line containing a comment because the presence of the apostrophe at the start of the comment will cause the balance of the line (including the code) to be ignored.

2.7 Breaking a BasicScript Statement across Multiple Lines

By default, in Script Editor, a single BasicScript statement can extend only as far as the right margin, and each line break represents a new statement. However, you can override this default if you want to break a long statement across two or more lines.

Here's how to indicate that two or more lines of BasicScript code should be treated as a single statement when your script is run.

To break a BasicScript statement across multiple lines

1. Type the BasicScript statement on multiple lines, exactly the way you want it to appear.
2. Place the insertion point at the end of the first line in the series.
3. Press the spacebar once to insert a single space.
4. Type an underscore (_).
   NOTE: the underscore is the *line continuation character*, which indicates that the BasicScript statement continues on the following line.
5. Repeat steps 2-4 to place a line-continuation character at the end of each line in the series except the last.

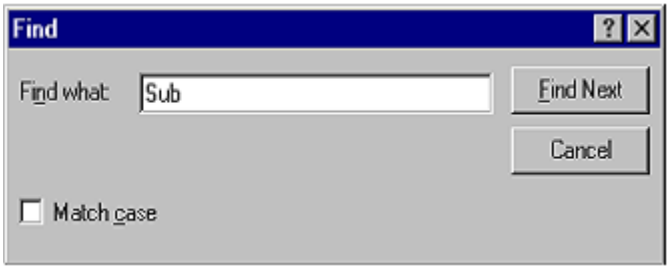
Attention
When you run your script, the code on this series of lines will be executed as a single BasicScript statement, just as if you had typed the entire statement on the same line.

2.8 Searching and Replacing

Script Editor makes it easy to search for specified text in your script and automatically replace instances of specified text.

To find specified text

1. Move the insertion point to where you want to start your search. (To start at the beginning of your script, press Ctrl+Home.)
2. Press Ctrl+F. Script Editor displays the Find dialog box

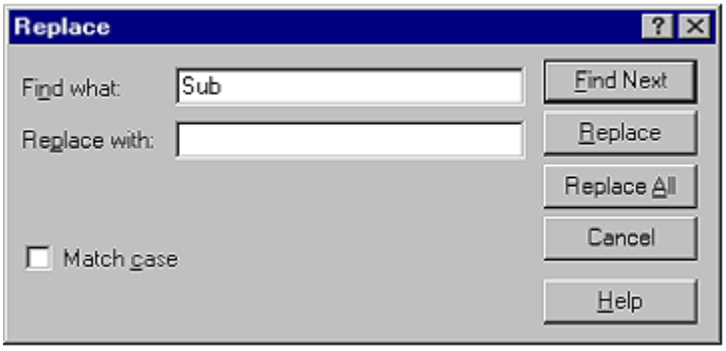


3. In the Find What field, specify the text you want to find.
4. Select the Match Case check box if you want the search to be case-sensitive. Otherwise, the search will be case-insensitive.
5. Click the Find Next button or press Enter.
   The Find dialog box remains displayed, and Script Editor either highlights the first instance of the specified text or indicates that it cannot be found.
6. If the specified text has been found, repeat step 5 to search for the next instance of it.

> **Attention**
> If the Find dialog box blocks your view of an instance of the specified text, you can move the dialog box out of your way and continue with your search. You can also click the Cancel button, which removes the Find dialog box while maintaining the established search criteria, and then press F3 to find successive occurrences of the specified text. (If you press F3 when you have not previously specified text for which you want to search, Script Editor displays the Find dialog box so you can specify the desired text.).

To replace specified text

1. Move the insertion point to where you want to start the replacement operation. (To start at the beginning of your script, press Ctrl+Home.)
2. Choose the Replace command from the Search menu.
   Script Editor displays the Replace dialog box:



3. In the Find What field, specify the text you want to replace.

4. In the Replace With field, specify the replacement text.
5. Select the Match Case check box if you want the replacement operation to be case-sensitive. Otherwise, it will be case-insensitive.
6. To replace all instances of the specified text, click the Replace All button.

   Script Editor either replaces the specified text throughout your script or indicates the number of occurrences it has changed, or it indicates that the specified text cannot be found.
7. To replace selected instances of the specified text, click the Find Next button.

   Script Editor either highlights the first instance of the specified text or indicates that it cannot be found.
8. If the specified text has been found, either click the Replace button to replace that instance of it or click the Find Next button to highlight the next instance (if any).

   Each time you click the Replace button; Script Editor replaces that instance of the specified text and automatically highlights the next instance.

2.9 Checking the Syntax of a Script

When you try to run or debug a script whose syntax hasn't been checked, Script Editor first performs a syntax check automatically.

1. From the Run menu, choose the Syntax Check command.
   Script Editor either indicates that no errors have been found or displays an error message that specifies the first line in your script where an error has been found and briefly describes the nature of that error.
2. Click the OK button or press Enter.
   If Script Editor has found a syntax error, the line containing the error is highlighted on your display.
3. Correct the syntax error.
4. Repeat steps 1-3 until you have found and corrected all syntax errors.

2.10 Editing Dialog Box Templates

The Insert New Dialog and Edit Dialog commands appear in the Edit menu. These commands allow you to use the features of Dialog Editor to create a new dialog box template and insert it into your script or edit an existing dialog box template contained in your script.

To insert a new dialog box template into your script

1. Place the insertion point where you want the new dialog box template to appear in your script.
2. From the Edit menu, choose the Insert New Dialog command.
   Script Editor's application window is temporarily disabled, and Dialog Editor appears, displaying a new dialog box in its application window.
3. Use Dialog Editor to create your dialog box.
4. Exit from Dialog Editor and return to Script Editor.
   Script Editor automatically places the new dialog box template generated by Dialog Editor in your script at the location of the insertion point.

To edit an existing dialog box template in your script

1. Select the BasicScript code for the entire dialog box template.
2. From the Edit menu, choose the Edit Dialog command.
   Script Editor's application window is temporarily disabled, and Dialog Editor appears, displaying in its application window a dialog box created from the code you selected.
3. Use Dialog Editor to modify your dialog box.
4. Exit from Dialog Editor and return to Script Editor. Script Editor automatically replaces the dialog box template you originally selected with the revised template generated by Dialog Editor.
   Refer to “Recording Scripts” for a detailed discussion of how to use Dialog Editor to create and edit dialog box templates.

2.11 Running Your Scripts

Once you have finished editing your script, you will want to run it to make sure it performs the way you intended. You can also pause or stop an executing script.

Here's how to compile your script, if necessary, and then execute it.

To run your script

1. Click the Start tool on the toolbar.
2. -Or-
3. Press F5.
4. The script is compiled (if it has not already been compiled), the focus is switched to the parent window, and the script is executed.

> **Attention**
> During script execution, Script Editor's application window is available only in a limited manner. Some of the menu commands may be disabled, and the toolbar tools may be inoperative.

To pause an executing script

- Press Ctrl+Break.
  Execution of the script is suspended, and the instruction pointer (a gray highlight) appears on the line of code where the script stopped executing.

> **Attention**
> The instruction pointer designates the line of code that will be executed next if you resume running your script.

To stop an executing script

- Click the End tool on the toolbar.

> **Attention**
> Many of the functions of Script Editor's application window may be unavailable while you are running a script. If you want to stop your script but find that the toolbar is currently inoperative, press Ctrl+Break to pause your script, then click the End tool.

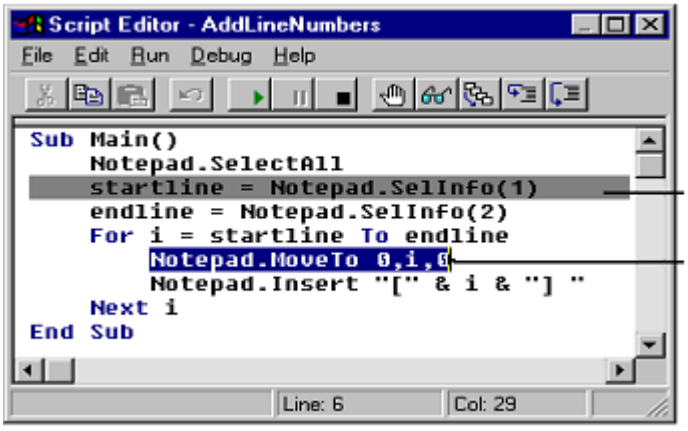
2.12 Debugging Your Scripts

This section presents some general information that will help you work most effectively with Script Editor's debugging capabilities and then explains how to trace the execution of your script, how to set and remove breakpoints, and how to add watch variables and modify their value.

2.12.1 Using the BasicScript Debugger

While debugging, you are actually executing the code in your script line by line. Therefore, to prevent any modifications to your script while it is being run, the edit pane is read-only during the debugging process. You are free to move the insertion point throughout the script, select text and copy it to the Clipboard as necessary, set breakpoints, and add and remove watch variables, but you cannot make any changes to the script until you stop running it.

To let you follow and control the debugging process, Script Editor displays an *instruction pointer* on the line of code that is about to be executed\that is, the line that will be executed next if you either proceed with the debugging process or run your script at full speed. When the instruction pointer is on a line of code, the text on that line appears in black on a gray background that spans the width of the entire line. The following illustration shows the difference between the instruction pointer and the selection highlight (discussed in the preceding section), in which the text appears in white on a black background that spans only the width of the selected text.



2.13 Tracing Script Execution

Script Editor gives you two ways to trace script execution\single step and procedure step\both of which involve stepping through your script code line by line. The distinction between the two is that the single step process traces into calls to user-defined functions and subroutines, whereas the procedure step process does not trace into these calls (although it does execute them).

When you are stepping through a subroutine, you may need to determine the procedure calls by which you arrived at that point in your script. This section also describes how to use the Calls dialog box to obtain this information.

When you are stepping through a subroutine, you may want to repeat or skip execution of a section of code.

This section describes the procedures for using the Calls dialog box and also for using the Set Next Statement command to move the instruction pointer to another line within that subroutine.

Here's how to trace the execution of your script with either the single step or procedure step method.

Tip

When you initiate execution of your script with any of these methods, the script will first be compiled, if necessary. Therefore, there may be a slight pause before execution actually begins. If your script contains any compile errors, it will not be executed. To debug your script, first correct any compile errors, and then initiate execution again.

To step through your script

1. Click the Single Step or Procedure Step tool on the toolbar or
   Press F8 (Single Step) or Shift+F8 (Procedure Step).

   Script Editor places the instruction pointer on the Sub Main line of your script.
2. To continue tracing the execution of your script line by line, repeat step 1.
   Each time you repeat step 1, Script Editor executes the line containing the instruction pointer and moves the instruction pointer to the next line to be executed.
3. When you finish tracing the execution of your script, either click the Start tool on the toolbar (or press F5) to run the balance of the script at full speed or click the End tool to halt execution of the script.

To display the Calls dialog box

1. Click the Calls tool on the toolbar.
   Script Editor displays the Calls dialog box, which lists the procedure calls made by your script in the course of arriving at the present subroutine.
2. From the Calls dialog box, select the name of the procedure you wish to view.
3. Click the Show button.
   Script Editor highlights the currently executing line in the procedure you selected, scrolling that line into view if necessary. (During this process, the instruction pointer remains in its original location in the subroutine.

To move the instruction pointer to another line within a subroutine

1. Place the insertion point in the line where you want to resume stepping through the script.
2. From the Debug menu, choose the Set Next Statement command. The instruction pointer moves to the line you selected, and you can resume stepping through your script from there.

Tip

You can only use the Set Next Statement command to move the instruction pointer within the same subroutine. If you place the insertion point on a line that is not in the same subroutine, the Set Next Statement command will be disabled in the Debug menu.

2.14 Setting and Removing Breakpoints

If you want to start the debugging process at the first line of your script and then step through your code line by line until you reach the end of the code that you need to debug, the method described in the preceding subsection works fine. But if you only need to debug one or more portions of a long script, that method can be pretty cumbersome.

An alternate strategy is to set one or more *breakpoints* at selected lines in your script. Script Editor suspends execution of your script just before it reaches a line containing a breakpoint, thereby allowing you to begin or resume stepping through the script from that point.

2.14.1 Setting Breakpoints

You can set breakpoints to begin the debugging process partway through your script, to continue debugging at a line outside the current subroutine, and to debug only selected portions of your script.

Valid breakpoints can only be set on lines in your script that contain code, including lines in functions and subroutines. Although you can set a breakpoint anywhere within a script prior to execution, when you compile and run the script, invalid breakpoints (that is, breakpoints on lines that don't contain code) are automatically removed. While you are debugging your script, Script Editor will beep if you try to set a breakpoint on a line that does not contain code.

To start debugging partway through a script

1. Place the insertion point in the line where you want to start debugging.
2. To set a breakpoint on that line, click the Toggle Breakpoint tool on the toolbar.
   Or
   Press F9.
   The line on which you set the breakpoint now appears in contrasting type.
3. Click the Start tool on the toolbar
   Or
   Press F5.
   Script Editor runs your script at full speed from the beginning and then pauses prior to executing the line containing the breakpoint. It places the instruction pointer on that line to designate it as the line that will be executed next when you either proceed with debugging or resume running the script.
   If you want to continue debugging at another line in your script, you can use the Set Next Statement command in the Debug menu to move the instruction pointer to the desired line-provided the line is within the same subroutine.

To continue debugging at a line outside the current subroutine

1. Place the insertion point in the line where you want to continue debugging.
2. To set a breakpoint on that line, press F9.
3. To run your script, click the Start tool on the toolbar or press F5.
   The script executes at full speed until it reaches the line containing the breakpoint and then pauses with the instruction pointer on that line. You can now resume stepping through your script from that point.

To debug selected portions of your script

1. Place a breakpoint at the start of each portion of your script that you want to debug.
   Note: Up to 255 lines in your script can contain breakpoints.
2. To run the script, click the Start tool on the toolbar or press F5.

The script executes at full speed until it reaches the line containing the first breakpoint and then pauses with the instruction pointer on that line.

3 Step through as much of the code as you need to.

4 To resume running your script, click the Start tool on the toolbar or press F5.

The script executes at full speed until it reaches the line containing the second breakpoint and then pauses with the instruction pointer on that line.

5 Repeat steps 3 and 4 until you have finished debugging the selected portions of your script.

To manually remove a single breakpoint

1 Place the insertion point on the line containing the breakpoint that you want to remove.

2 Click the Toggle Breakpoint tool on the toolbar OR

Press F9.

The breakpoint is removed, and the line no longer appears in contrasting type.

To remove all breakpoints manually

- Select the Clear All Breakpoints command from the Debug menu.

Script Editor removes all breakpoints from your script.

Breakpoints are removed automatically under the following circumstances: (1) As mentioned earlier, when your script is compiled and executed, breakpoints are removed from lines that don't contain code. (2) When you exit from Script Editor, all breakpoints are cleared.

2.15 Adding a Watch Variable

As you debug your script, you can use Script Editor's watch pane to monitor selected variables. For each of the variables on this watch variable list, Script Editor displays its context, name, and value. The values of the variables on the watch list are updated each time you enter break mode.

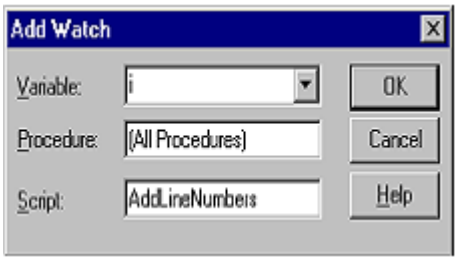
To add a watch variable

1 Click the Add Watch tool on the toolbar.

Or

Press Shift+F9

Script Editor displays the Add Watch dialog box.



2 In the Variable box, type or select the name of the variable you want to add to the watch variable list.

If you are executing the script, you can click the arrow in the Variable box to display the names of all variables that are "in scope," or defined within the current function or subroutine. You can then select the variable you want from the open list.

3 In the Procedure box, type or select the name of the procedure containing the variable you want to watch. If the variable you want to watch is a private or public variable, then type or select "(All Procedures)".

4 In the Script box, type or select the name of the script containing the variable you want to watch. If the variable you want to watch is a public variable, then type or select "(All Scripts)".

5 Click OK to add the variable to the watch variable list.

6 Script Editor displays the context, name, and value of the variable in a three-column list in the watch pane. If you have previously added other watch variables, Script Editor will display these as well.

2.15.1 Types of Variables You Can Watch

Script Editor permits you to monitor only variables of fundamental data types, such as Integer, Long, Variant, and so on; you cannot watch complex variables, such as structures or arrays, or expressions using arithmetic operators. You can, however, watch individual elements of arrays or structure members using the following syntax:

[*variable* [(*index*,...)] [.*member* [(*index*,...)]]...]

where *variable* is the name of the structure or array variable, *index* is a literal number, and *member* is the name of a structure member.

For example, the following are valid watch expressions:

Watch Variable	Description
a(1)	Element 1 of array a
person.age	Member age of structure person
company(10,23).person.age	Member age of structure person that is at element 10,23 within the array of structures called company

2.16 Modifying or Deleting a Watch Variable

In order to modify the value of a variable or remove the variable from the watch pane, you must first select the desired variable. When the debugger has control, you can modify the value of any of the variables on Script Editor's watch variable list.

To select a watch variable

1 Place the mouse pointer on the variable you want to select and click the left mouse button.

2 -Or-

3 If one of the variables on the watch list is already selected, use the arrow keys to move the selection highlight to the desired variable.

4 -Or-

5 If the insertion point is in the edit pane, press F6 to highlight the most recently selected variable on the watch list and then use the arrow keys to move the selection highlight to the desired variable.

> **Attention**
> Pressing F6 again returns the insertion point to its previous position in the edit pane.

To modify the value of a variable on the watch variable list

1 Place the mouse pointer on the name of the variable whose value you want to modify and double-click the left mouse button.

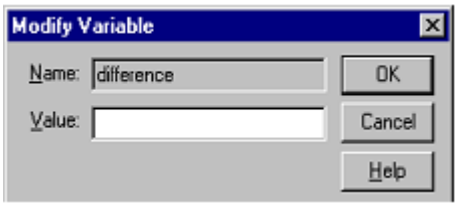
Or

Select the name of the variable whose value you want to modify and press Enter or F2.

Note: The name of the variable you selected on the watch variable list appears in the Name field. If you want to change another variable, you can either enter a different variable in the Name field or select a different variable from the Variables list box, which shows the names of the variables that are defined within the current function or subroutine.

When you use the Modify Variable dialog box to change the value of a variable, you don't have to specify the context. Script Editor first searches locally for the definition of that variable, then privately, then publicly.

Script Editor displays the Modify Variable dialog box.



2 Enter the new value for your variable in the Value field.

Click the OK button.

The new value of your variable appears on the watch variable list.

When changing the value of a variable, Script Editor will, if necessary, convert the value you entered to match the type of the variable. For example, if you change the value of an Integer variable to 1.7, Script Editor converts this value from a floating-point number to an Integer is performed, assigning the value 2 to the variable.

When modifying a Variant variable, Script Editor needs to determine both the type and value of the data. Script Editor uses the following logic in performing this assignment (in this order):

If the new value is	Then
Null	The Variant variable is assigned Null (VarType 1).
Empty	The Variant variable is assigned Empty (VarType 0)
True	The Variant variable is assigned True (VarType 11).
False	The Variant variable is assigned False (VarType 11).
number	The Variant variable is assigned the value of *number*. The type of the variant is the smallest data type that fully represents that number. You can force the data type of the variable by using a type-declaration letter following *number*, such as %, #, &, !, or @.
date	The Variant variable is assigned the value of the new date (VarType 7).
Anything else	The Variant variable is assigned a String (VarType 8)

Attention
Script Editor will not assign a new value if it cannot be converted to the same type as the specified variable.

To delete a watch variable

1 Select the variable on the watch list.

2 Press Del.
The selected variable is removed from the watch list.

2.17 Exiting From Script Editor

Here's how to get out of Script Editor. What happens when you exit depends on (1) whether you have made changes to your script and (2) whether your script contains errors.

To exit from Script Editor

1. Choose the Exit and Return command from the File menu.
2. If you *have* made changes to your script, Script Editor displays a dialog box asking whether you want to save the script. If you either click the No button or click the Yes button and your script contains no errors, you exit from Script Editor immediately. If you click the Yes button and your script contains errors, Script Editor highlights the line containing the first error and displays a dialog box asking whether you want to exit anyway. If you click the Yes button, Script Editor saves your script, errors and all, and then you exit from Script Editor.
3. If you *haven't* made any changes to your script, you exit from Script Editor immediately, regardless of whether the script contains errors from a previous editing session.

2.18 Menu Reference

File Menu

Command	Keyboard Shortcut	Function
E*x*it and Return	Esc	Compiles your script and returns you to the host application.

Edit Menu

Command	Keyboard Shortcut	Function
*U*ndo	Ctrl+Z	Reverses the effect of the preceding editing change(s).
Cu*t*	Ctrl+X	Removes the selected text from the script and places it on the Clipboard.
*C*opy	Ctrl+C	Copies the selected text, without removing it from the script, and places it on the Clipboard.
*P*aste	Ctrl+V	Inserts the contents of the Clipboard at the current position of the insertion point.
*D*elete	Del or BkSp	Removes the selected text from the script without placing it on the Clipboard.
*I*nsert New Dialog…		Invokes the Summit Dialog Editor, which you can use to create a new dialog box for insertion into your script
*E*dit Dialog…		Invokes the Summit Dialog Editor, which you can use to edit the selected dialog box template. (This command is only enabled if a dialog box template is currently selected.)
*F*ind…	Ctrl+F	Displays the Find dialog box, which allows you to specify text for which you want to search.
Find *N*ext	F3	Searches for the next occurrence of previously specified text. If you have not previously specified text for which you want to search, displays the Find dialog box.
*R*eplace…		Displays the Replace dialog box, which allows you to substitute replacement text for instances of specified text.
*G*oto Line…	Ctrl+G	Presents the Goto Line dialog box, which allows you to move the insertion point to the start of a specified line number in your script.

Tip
The Insert New Dialog and Edit Dialog commands only appear in the Edit menu if you are running the BasicScript 2.2 on a platform that supports Dialog Editor

Run Menu

Command	Keyboard Shortcut	Function
*S*tart	F5	Begins execution of script.
*E*nd		Stops execution of script
S*y*ntax Check		Verifies the syntax of the statements in your script by compiling it.

Debug Menu

Command	Keyboard Shortcut	Function
*A*dd Watch…	Shift+F9	Displays the Add Watch dialog box, in which you can specify the name of a BasicScript variable. That variable, together with its value (if any), is then displayed in the watch pane of Script Editor's application window.
*D*elete Watch	Del	Deletes a selected variable from the watch variable list.
*M*odify…	Enter or F2	Displays the Modify Variable dialog box for a selected variable, which enables you to modify the value of that variable.
*S*ingle Step	F8	Steps through the script code line by line, tracing into called procedures
*P*rocedure Step	Shift+F8	Steps through the script code line by line without tracing into called procedures.
*T*oggle Breakpoint	F9	Toggles a breakpoint on the line containing the insertion point
*C*lear All Breakpoints		Removes all breakpoints previously set with the Toggle Breakpoint command.
Set *N*ext Statement		Enables you to place the instruction pointer on another line within the current procedure and resume script execution from that point.

Help Menu

Command	Keyboard Shortcut	Function
*C*ontents		Displays a list of major topics on which you can obtain help
*S*earch for Help on…		Displays the Search dialog box, which allows you to search for Help topics containing specific keyword.

3 Editing Custom Dialog Boxes

Dialog Editor is a tool that enables you to create and modify custom dialog boxes for use in your BasicScript scripts. Although the BasicScript statements used to display a dialog box and respond to the choices made by a user of the dialog box may seem complicated, Dialog Editor makes it easy to generate the BasicScript statements needed for your custom dialog boxes. This section provides the following information about editing custom dialog boxes.

- Overview of Dialog Editor
- Using the Summit Dialog Editor
- Creating a Custom Dialog Box
- Editing a Custom Dialog Box
- Editing an Existing Dialog Box
- Testing an Edited Dialog Box
- Incorporating a Dialog Box Template into Your Script
- Exiting from Dialog Editor
- Using a Custom Dialog Box in Your Script
- Using a Dynamic Dialog Box in Your Script
- Menu Reference

3.1 Overview of Dialog Editor

Sometimes your script will need to obtain information from the user. In many cases, you can obtain this information by using one of BasicScript's predefined dialog boxes in your script. When you must go beyond the information-gathering capabilities provided by predefined dialog boxes, you can use Dialog Editor to create a custom dialog box for use in your script.

Dialog Editor is a tool that allows you to generate a *dialog box template* in BasicScript simply by editing an on-screen dialog box layout. You can then incorporate the template that Dialog Editor generates into your script. The balance of this section provides general information that you'll need in order to work with Dialog Editor, including:

- Features that Dialog Editor supports
- An introduction to Dialog Editor's application window
- A list of keyboard shortcuts
- How to use the Help system

Then, in the following sections, you'll learn how to use Dialog Editor to create and edit custom dialog boxes and to edit dialog boxes captured from other applications. You'll also learn how to test an edited dialog box and incorporate the dialog box template generated by Dialog Editor into your script. And finally, you'll learn how to exit from Dialog Editor.

3.1.1 Features of the Summit Dialog Editor

Dialog Editor supports the following features:

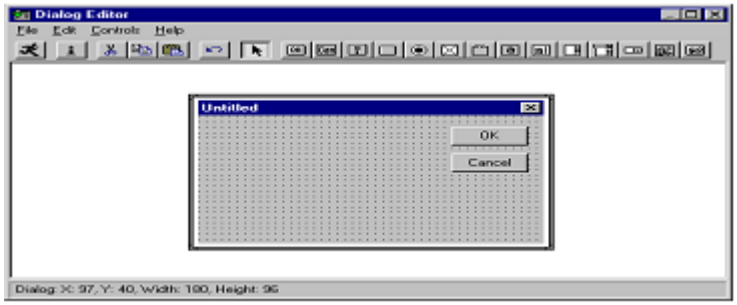
- Visual editing of a dialog box template in BasicScript
- The creation of dynamic dialog boxes

3.2 Using the Summit Dialog Editor

This section presents general information that will help you work most effectively with Dialog Editor. It includes an overview of Dialog Editor's application window— the interface you'll use to create and edit dialog box templates in BasicScript— as well as a list of keyboard shortcuts and information on using the Help system.

3.3 Dialog Editor's Application Window

Before you begin creating a new custom dialog box, Dialog Editor's application window looks like this:



The application window contains the following elements:

- Toolbar: a collection of tools that you can use to provide instructions to Dialog Editor, as discussed in the following subsection
- Dialog box: the visual layout of the dialog box that you are currently creating or editing
- Status bar: provides key information about the operation you are currently performing, including the name of the currently selected control or dialog box, together with its position on the display and its dimensions; the name of a control you are about to add to the dialog box with the mouse pointer, together with the pointer's position on the display; the function of the currently selected menu command; and the activation of Dialog Editor's testing or capturing functions

Attention

Dialog boxes created with Dialog Editor normally appear in an 8 point Helvetica font, both in Dialog Editor's application window and when the corresponding BasicScript code is run.

3.3.1 Dialog Editor's Toolbar

The following list briefly explains the purpose of each of the tools on Dialog Editor's toolbar, which you can use to crate and test dialogs.

Icon	Tool	Function
	Run	Runs the dialog box for testing purposes.
	Information	Displays information for the selected control.
	Cut	Removes the selected control from the dialog box.
	Copy	Copies the selected control to the clipboard.
	Paste	Inserts the clipboard into the active dialog box.
	Undo	Reverses the effect of the preceding editing change(s).

Icon	Tool	Function
	Pick	Lets you select, move, and resize items and control the insertion point.
	OK Button	Adds an OK button to your dialog box.
	Cancel Button	Adds a Cancel button to your dialog box.
	Help Button	Adds a Help button to your dialog box.
	Push Button	Adds a push button to your dialog box.
	Option Button	Adds an option button to your dialog box.
	Check Box	Adds a check box to your dialog box.
	Group Box	Adds a group box to your dialog box.
	Text	Adds a text control to your dialog box.
	Text Box	Adds a text box to your dialog box.
	List Box	Adds a list box to your dialog box.
	Combo Box	Adds a combo box to your dialog box.
	Drop List Box	Adds a drop list box to your dialog box.
	Picture	Adds a picture to your dialog box.
	Picture Button	Adds a picture button to your dialog box.

3.4 Keyboard Shortcuts for Dialog Editor

The following keyboard shortcuts can be used for some of the operations you will perform most frequently in Dialog Editor.

Key(s)	Function
Alt+F4	Closes Dialog Editor's application window.
Ctrl+C	Copies the selected dialog box or control, without removing it from Dialog Editor's application window, and places it on the Clipboard.
Ctrl+D	Creates a duplicate copy of the selected control.
Ctrl+G	Displays the Grid dialog box.
Ctrl+I	Displays the Information dialog box for the selected dialog box or control.
Ctrl+V	Inserts the contents of the Clipboard into Dialog Editor. If the Clipboard contains BasicScript statements describing one or more controls, then Dialog Editor adds those controls to the current dialog box. If the Clipboard contains the BasicScript template for an entire dialog box, then Dialog Editor creates a new dialog box from the statements in the template.
Ctrl+X	Removes the selected dialog box or control from Dialog Editor's application window and places it on the Clipboard.
Ctrl+	Undoes the preceding operation.
Del	Removes the selected dialog box or control from Dialog Editor's application window without placing it on the Clipboard.
F1	Displays Help for the currently active window.
F2	Sizes certain controls to fit the text they contain.
F5	Runs the dialog box for testing purposes.
Shift+F1	Toggles the Help pointer.

3.5 Using the Help System

Dialog Editor provides several ways to obtain on-line help.

To display Help for the currently active window

- Press F1.
  If Dialog Editor's application window was active, the Help system contents appear. If a dialog box was active, Help for that dialog box appears

To pinpoint a specific topic in the Help system

1. From the Help menu, choose the Search for Help on command.
   A scrollable list of Help topics appears.
2. Select the desired topic from the list.
   The topic you selected is displayed in a second scrollable list, together with closely related Help topics, if any
3. If the desired topic is not already highlighted on the second list, select it and press Enter.
   Help is displayed for the topic you selected.

3.6 Creating a Custom Dialog Box

This section describes the types of controls that Dialog Editor supports. It also explains how to create controls and initially position them within your dialog box, and offers some pointers on creating controls efficiently.

In the next section, Editing a Custom Dialog Box, you will learn how to make various types of changes to the controls that you have created-moving and resizing them, assigning labels and accelerator keys, and so forth.

3.7 Types of Controls

Dialog Editor supports the following types of standard Windows controls:

Push button

This is a command button. The OK, Cancel, and Help buttons are special types of push buttons.

Option button

This button is a group of two or more linked buttons. This allows the user to select only one from a group of mutually exclusive choices. A group of option buttons works the same way as the buttons on a car radio: because the buttons operate together as a group, clicking an unselected button in the group selects that button and automatically deselects the previously selected button in that group.

Check box

A box that users can check or clear to indicate their preference regarding the alternative specified on the check box label.

Group box

A rectangular design element used to enclose a group of related controls. You can use the optional group box label to display a title for the controls in the box.

Text

A field containing text that you want to display for the users' information. The text in this field wraps, and the field can contain a maximum of 255 characters. Text controls can either display stand-alone text or be used as labels for text boxes, list boxes, combo boxes, drop list boxes, pictures, and picture buttons. You can choose the font in which the text appears.

Text box

This is a field into which you can enter text (potentially, as much as 32K). By default, this field holds a single line of nonwrapping text. If you choose the Multiline setting in the Text Box Information dialog box, this field will hold multiple lines of wrapping text.

List box

This is a list of items from which users can select one item. User can scroll through the list. The selected item is highlighted on the list.

Combo box

This is a text field with a list of items for selection. Users can either select an item from the list or enter the name of the desired item in the text field. User can scroll through the list.The selected item is displayed in the text field. If the item was selected from the scrolling list, it is highlighted there as well.

Drop list box

A field that displays the currently selected item, followed by a downward-pointing arrow, which users can click to temporarily display a scrolling list of items. Once they select an item from the list, the list disappears and the newly selected item is displayed in the field.

Picture

A field used for displaying a Windows bitmap or metafile.

Picture button

A special type of push, or command, button on which a Windows bitmap or metafile appears.

Tip

Group boxes, text controls, and pictures are passive elements in a dialog box, inasmuch as they are used purely for decorative or informative purposes. Users cannot act upon these controls, and when they tab through the dialog box, the focus skips over these controls.

You can obtain a Windows bitmap or metafile from a file or from a specified library.

3.8 Adding Controls to a Dialog Box

In this subsection, you'll learn how to create controls and determine approximately where they first appear within your dialog box. In the following subsection, you'll learn how to determine the positioning of controls more precisely.

To add a control

1 Here's how to add one or more controls to your dialog box using simple mouse and keyboard methods.

2 From the toolbar, choose the tool corresponding to the type of control you want to add.

When you pass the mouse pointer over an area of the display where a control can be placed, the pointer becomes an image of the selected control with crosshairs (for positioning purposes) to its upper left. The name and position of the selected control appear on the status bar. When you pass the pointer over an area of the display where a control cannot be placed, the pointer changes into a circle with a slash through it (the “prohibited” symbol).

Note: You can only insert a control within the borders of the dialog box you are creating. You cannot insert a control on the dialog box's title bar or outside its borders.

3 Place the pointer where you want the control to be positioned and click the mouse button.

The control you just created appears at the specified location. (To be more specific, the upper left corner of the control will correspond to the position of the pointer's crosshairs at the moment you clicked the mouse button.) The control is surrounded by a thick frame, which means that it is selected, and it may also have a default label.

After the new control has appeared, the mouse pointer becomes an arrow, to indicate that the Pick tool is active and you can once again select any of the controls in your dialog box.

To add another control of the same type as the one you just added

- Press Ctrl+D. A duplicate copill y of the control appears.

To add a different type of control

- Repeat steps 1 and 2 above.

To reactivate the Pick tool

- Click the arrow-shaped tool on the toolbar.

  -Or-

  Place the mouse pointer on the title bar of the dialog box or outside the borders of the dialog box (that is, on any area where the mouse pointer turns into the “prohibited” symbol) and click the mouse button.

  As you plan your dialog box, keep in mind that a single dialog box can contain no more than 255 controls and that a dialog box will not operate properly unless it contains either an OK button, a Cancel button, a push button, or a picture button. (When you create a new custom dialog box, an OK button and a Cancel button are provided for you by default.)

  Later in the chapter, you'll learn more about selecting controls, and you'll learn how to assign labels.

3.9 Using the Grid to Help You Position Controls within a Dialog Box

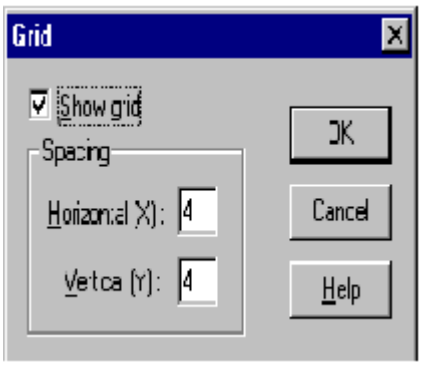
The preceding subsection explained how to determine approximately where a newly created control will materialize in your dialog box. Here, you'll learn how to use Dialog Editor's grid to help you fine-tune the initial placement of controls.

The area of your dialog box in which controls can be placed (that is, the portion of the dialog box below the title bar) can be thought of as a grid, with the X (horizontal) axis and the Y (vertical) axis intersecting in the upper left corner (the 0, 0 coordinates). The position of controls can be expressed in terms of X units with respect to the left border of this area and in terms of Y units with respect to the top border. (In fact, the position of controls is expressed in this manner within the dialog box template that you produce by working with Dialog Editor.)

To display and adjust the grid

Here's how to display the grid and adjust its X and Y settings, which can help you position controls more precisely within your dialog box.

1. Press Ctrl+G.



   Dialog Editor displays the following dialog box
2. To display the grid in your dialog box, select the Show grid check box.
3. To change the current X and Y settings, enter new values in the X and Y fields.

   Note; The values of X and Y in the Grid dialog box determine the grid's spacing. Assigning smaller X and Y values produces a more closely spaced grid, which enables you to move the mouse pointer in smaller horizontal and vertical increments as you position controls. Assigning larger X and Y values produces the opposite effect on both the grid's spacing and the movement of the mouse pointer. The X and Y settings entered in the Grid dialog box remain in effect regardless of whether you choose to display the grid.
4. Click the OK button or press Enter.

   Dialog Editor displays the grid with the settings you specified. With the grid displayed, you can line up the crosshairs on the mouse pointer with the dots on the grid to position controls precisely and align them with respect to other controls.

As you move the mouse pointer over the dialog box after you have chosen a control tool from the toolbar, the status bar displays the name of the type of control you have selected and continually updates the position of the mouse pointer in X and Y units. (This information disappears if you move the mouse pointer over an area of the screen where a control cannot be placed.) After you click the mouse button to add a control, that control remains selected, and the status bar displays the control's width and height in dialog units as well as its name and position, as shown in the preceding illustration, in which the push button is selected.

Attention

Dialog units represent increments of the font in which Dialog Editor creates dialog boxes (namely, 8 point Helvetica). Each X unit represents an increment equal to \ of that font, and each Y unit represents an increment equal to 1/8 of that font.

3.10 Creating Controls Efficiently

Creating dialog box controls in random order might seem like the fastest approach. However, the order in which you create controls has some important implications, so a little advance planning can save you a lot of work in the long run.

Here are several points about creating controls that you should keep in mind.

Tabbing order

Users can select dialog box controls by tabbing, as explained in the next subsection. The order in which you create the controls is what determines the tabbing order. That is, as users tab through the dialog box, the focus is changed from one control to the next in the order in which you created the controls (regardless of the order in which you position the controls in the dialog box). The closer you can come to creating controls in the order in which you want them to receive the tabbing focus, the fewer tabbing-order adjustments you'll have to make later on.

Option button grouping

If you want a series of option buttons to work together as a mutually exclusive group, you must create all the buttons in that group one right after the other, in an unbroken sequence. If you get sidetracked and create a different type of control before you have finished creating all the option buttons in your group, you'll split the buttons into two (or more) separate groups. (Let's say you want to create an option button group with five buttons. You create three of the buttons and then create a list box, after which you finish creating the last two buttons. When you test your dialog box, you'll find that all five of these option buttons don't work together as a mutually exclusive group. Instead, the first three buttons will form one mutually exclusive group, and the last two buttons will form another mutually exclusive group.)

Accelerator keys

As explained later in the chapter, you can provide easy access to a text box, list box, combo box, or drop list box by assigning an accelerator key to an associated text control, and you can provide easy access to the controls in a group box by assigning an accelerator key to the group box label. To do this, you must create the text control or group box first, followed immediately by the controls that you want to associate with it. If the controls are not created in the correct order, they will not be associated in your dialog box template and any accelerator key you assign to the text control or group box label will not work properly.

If you don't create controls in the most efficient order, the resulting problems with tabbing order, option button grouping, and accelerator keys usually won't become apparent until you test your dialog box. Although you can still fix these problems at that point, as explained later in the chapter, it will definitely be more cumbersome. In short, it's easier to prevent (or at least minimize) problems of this sort than to fix them after the fact.

3.11 Editing a Custom Dialog Box

In the preceding section, you learned how to create controls and determine where they initially appear within your dialog box. In this section, you'll learn how to make various types of changes to both the dialog box and the controls in it. The following topics are included:

- Selecting items so that you can work with them
- Using the Information dialog box to check and/or change various attributes of items
- Changing the position and size of items
- Changing titles and labels
- Assigning accelerator keys
- Specifying pictures
- Creating or modifying picture libraries under Windows
- Duplicating and deleting controls
- Undoing editing operations

3.12 Selecting Items

In order to edit a dialog box or a control, you must first select it. When you select an item, it becomes surrounded by a thick frame, as you saw in the preceding section.

To select a control

1. With the Pick tool active, place the mouse pointer on the desired control and click the mouse button. Or With the Pick tool active, press the Tab key repeatedly until the focus moves to the desired control.
2. The control is now surrounded by a thick frame to indicate that it is selected and you can edit it.

To select the dialog box:

1. With the Pick tool active, place the mouse pointer on the title bar of the dialog box or on an empty area within the borders of the dialog box (that is, on an area where there are no controls) and click the mouse button Or With the Pick tool active, press the Tab key repeatedly until the focus moves to the dialog box.With the Pick tool active, press the Tab key repeatedly until the focus moves to the dialog box.
2. The dialog box is now surrounded by a thick frame to indicate that it is selected and you can edit it.

3.13 Using the Information Dialog Box

The Information dialog box enables you to check and adjust various attributes of controls and dialog boxes. This subsection explains how to display the Information dialog box and provides an overview of the attributes with which it lets you work. In the following subsections, you'll learn more about how to use the Information dialog box to make changes to your dialog box and its controls.

To display the Information dialog box for a dialog box

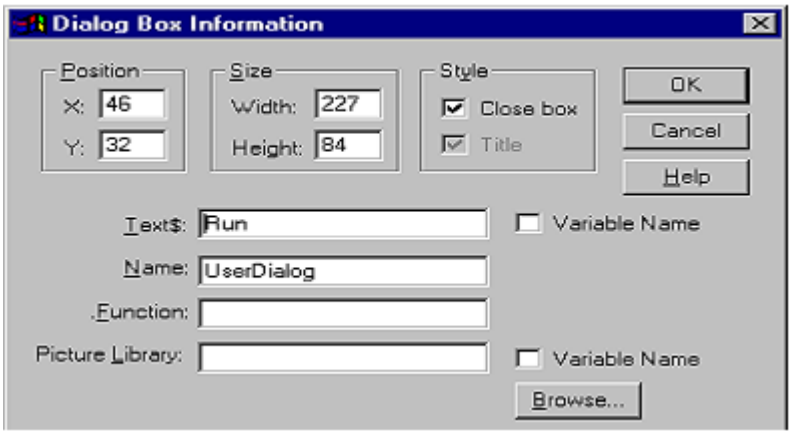
Here's how to use the Dialog Box Information dialog box to check and adjust attributes that pertain to the dialog box as a whole.

- With the Pick tool active, place the mouse pointer on an area of the dialog box where there are no controls and double-click the mouse button.

-Or-

With the Pick tool active, select the dialog box and either click the Information tool on the toolbar, press Enter, or press Ctrl+I.

Dialog Editor displays the Dialog Box Information dialog box.



To display the Information dialog box for a control

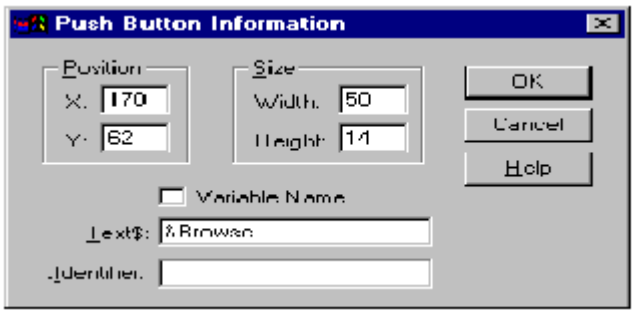
Here's how to use the Information dialog box for a control to check and adjust attributes that pertain to that particular control.

- With the Pick tool active, place the mouse pointer on the desired control and double-click the mouse button.

-Or-

With the Pick tool active, select the control and either click the Information tool on the toolbar, press Enter, or press Ctrl+I.

Dialog Editor displays an Information dialog box corresponding to the control you selected. Here is an example:



The following lists show the attributes that you can change with the Dialog Box Information dialog box and the Information dialog boxes for the various controls. In some cases (specified below), it's mandatory to fill in the fields in which the attributes are specified—that is, you must either leave the default information in these fields or replace it with more meaningful information, but you can't leave the fields empty. In other cases, filling in these fields is optional—that is, you can either leave the default information in the fields, replace it with more meaningful information, or leave the fields entirely empty.

Tip

A quick way to determine whether it's mandatory to fill in a particular Information dialog box field is to see whether the OK button becomes grayed out when you delete the information in that field. If it does, then you must fill in that field.

In many cases, you could simply leave the generic-sounding default information in the Information dialog box fields and worry about replacing it with more meaningful information after you paste the dialog box template into your script. However, if you take a few moments to replace the default information with something specific when you first create your dialog box, not only will you save yourself some work later on but you may also find that your changes make the BasicScript code produced by Dialog Editor more readily comprehensible and hence easier to work with.

3.13.1 Attributes That You Can Adjust with the Information Dialog Box

The Information Dialog Box can be used for checking and adjusting the following attributes, which pertain to the dialog box as a whole.

Mandatory/ Optional	Attribute
Optional	**Position:** X and Y coordinates on the display, in dialog units.
Mandatory	**Size:** width and height of the dialog box, in dialog units.
Optional	**Style:** options that allow you to determine whether the close box and title bar are displayed.
Optional	**Text$:** text displayed on the title bar of the dialog box.
Mandatory	**Name:** name by which you refer to this dialog box template in your BasicScript code.
Optional	.**Function:** name of a BasicScript function in your dialog box.
Optional	**Picture Library:** picture library from which one or more pictures in the dialog box are obtained.

3.13.2 Attributes That You Can Adjust with the Information Dialog Box for a Control

The Information dialog box for a control can be used to check and adjust the following attributes. The second column of the list indicates the control(s) to which each attribute pertains.

Mandatory/ Optional	Control(s) Affected	Attribute
Mandatory	All controls	**Position:** X and Y coordinates within the dialog box, in dialog units.
Mandatory	All controls	**Size:** width and height of the control, in dialog units.
Optional	Push button, option button, check box, group box, and text	**Text$:** text displayed on a control.
Optional	Help button	**FileName$:** name of the help file invoked when the user clicks this button.
Optional	Text	**Font:** font in which text is displayed.
Optional	Text box	**Multiline:** option that allows you to determine whether users can enter a single line of text or multiple lines.

Mandatory/ Optional	Control(s) Affected	Attribute
Optional	OK button, Cancel button, push button, option button, group box, text, picture, and picture button	.**Identifier:** name by which you refer to a control in your BasicScript code.
Mandatory	Check box, text box, list box, combo box, drop list box, and help button	.**Identifier:** name by which you refer to a control in your BasicScript code; also contains the result of the control after the dialog box has been processed.
Optional	Picture, picture button	.**Identifier:** name of the file containing a picture that you want to display or the name of a picture that you want to display from a specified picture library.
Optional	Picture	**Frame:** option that allows you to display a 3-D frame.
Mandatory	List box, combo box, and drop list box	**Array$:** name of an array variable in your BasicScript code
Mandatory	Option button	.**Option Group:** name by which you refer to a group of option buttons in your BasicScript code.

3.14 Changing Position and Size

This subsection explains how Dialog Editor helps you keep track of the location and dimensions of dialog boxes and controls, and presents several ways to move and resize these items.

3.14.1 Keeping Track of Position and Size

Dialog Editor's display can be thought of as a grid, in which the X (horizontal) axis and the Y (vertical) axis intersect in the upper left corner of the display (the 0, 0 coordinates). The position of the dialog box you are creating can be expressed in terms of X units with respect to the left border of the parent window and in terms of Y units with respect to the top border.

As explained in the preceding section, the portion of your dialog box below the title bar can also be thought of as a grid, with the X and Y axes intersecting in the upper left corner of this area. The position of controls within the dialog box can be expressed in terms of X units with respect to the left border of this area and in terms of Y units with respect to the top border.

When you select a dialog box or control, the status bar displays its position in X and Y units as well as its width and height in dialog units. Each time you move or resize an item, the corresponding information on the status bar is updated. You can use this information to position and size items more precisely.

3.14.2 Changing the Position of an Item

Dialog Editor provides several ways to reposition dialog boxes and controls.

To reposition an item with the mouse

1 With the Pick tool active, place the mouse pointer on an empty area of the dialog box or on a control.

2 Depress the mouse button and drag the dialog box or control to the desired location.

> **Attention**
> The increments by which you can move a control with the mouse are governed by the grid setting. For example, if the grid's X setting is 4 and its Y setting is 6, you'll be able to move the control horizontally only in increments of 4 X units and vertically only in increments of 6 Y units. This feature is handy if you're trying to align controls in your dialog box. If you want to move controls in smaller or larger increments, press Ctrl+G to display the Grid dialog box and adjust the X and Y settings

To reposition an item with the arrow keys

1 Select the dialog box or control that you want to move

2 Press an arrow key once to move the item by 1 X or Y unit in the desired direction.
Or

Depress an arrow key to “nudge” the item steadily along in the desired direction.

> **Attention**
> When you reposition an item with the arrow keys, a faint, partial afterimage of the item may remain visible in the item's original position. These afterimages are rare and will disappear once you test your dialog box

To reposition a dialog box with the Dialog Box Information dialog box

1 Display the Dialog Box Information dialog box.

2 Change the X and Y coordinates in the Position group box.
Or

Leave the X and/or Y coordinates blank

3 Click the OK button or press Enter

If you specified X and Y coordinates, the dialog box moves to that position. If you left the X coordinate blank, the dialog box will be centered horizontally relative to the parent window of the dialog box when the dialog box is run. If you left the Y coordinate blank, the dialog box will be centered vertically relative to the parent window of the dialog box when the dialog box is run.

To reposition a control with the Information dialog box

1 Display the Information dialog box for the control that you want to move.

2 Change the X and Y coordinates in the Position group box.

3 Click the OK button or press Enter.

The control moves to the specified position.

Tip

When you move a dialog box or control with the arrow keys or with the Information dialog box, the item's movement is not restricted to the increments specified in the grid setting.

When you attempt to test a dialog box containing hidden controls (i.e., controls positioned entirely outside the current borders of your dialog box), Dialog Editor displays a message advising you that there are controls outside the dialog box's borders and asks whether you wish to proceed with the test. If you proceed, the hidden controls will be disabled for testing purposes. (Testing dialog boxes is discussed later in the chapter.).

3.14.3 Changing the Size of an Item

Dialog boxes and controls can be resized either by directly manipulating them with the mouse or by using the Information dialog box. Certain controls can also be resized automatically to fit the text displayed on them.

To resize an item with the mouse

1 Here's how to change the size of a selected dialog box or control by dragging its borders or corners with the mouse.

2 With the Pick tool active, select the dialog box or control that you want to resize.

3 Place the mouse pointer over a border or corner of the item.

4 Depress the mouse button and drag the border or corner until the item reaches the desired size.

To resize an item with the Information dialog box

1 Here's how to change the size of a selected dialog box or control by changing its Width and/or Height settings in the Information dialog box.

2 Display the Information dialog box for the dialog box or control that you want to resize.

3 Change the Width and Height settings in the Size group box.

4 Click the OK button or press Enter.

The dialog box or control is resized to the dimensions you specified.

To resize selected controls automatically

1 Here's how to adjust the borders of certain controls automatically to fit the text displayed on them.

2 With the Pick tool active, select the option button, text control, push button, check box, or text box that you want to resize

3 Press F5.

The borders of the control will expand or contract to fit the text displayed on it

Attention

Windows metafiles always expand or contract proportionally to fit within the picture control or picture button control containing them. In contrast, Windows bitmaps are of a fixed size. If you place a bitmap in a control that is smaller than the bitmap, the bitmap is clipped off on the right and bottom. If you place a bitmap in a control that is larger than the bitmap, the bitmap is centered within the borders of the control. Picture controls and picture button controls must be resized manually.

3.15 Changing Titles and Labels

By default, when you begin creating a dialog box, its title reads “Untitled,” and when you first create group boxes, option buttons, push buttons, text controls, and check boxes, they have generic-sounding default labels, such as “Group Box” and “Option Button.”

To change a dialog box title or a control label

1. Here's how to change the title of your dialog box as well as the labels of group boxes, option buttons, push buttons, text controls, and check boxes.
2. Display the Information dialog box for the dialog box whose title you want to change or for the control whose label you want to change.
3. Enter the new title or label in the Text$ field.

   Note: Dialog box titles and control labels are optional. Therefore, you can leave the Text$ field blank.
4. If the information in the Text$ field should be interpreted as a variable name rather than a literal string, select the Variable Name check box.
5. Click the OK button or press Enter.

   The new title or label is now displayed on the title bar or on the control.

   Although OK and Cancel buttons also have labels, you cannot change them. The remaining controls (text boxes, list boxes, combo boxes, drop list boxes, pictures, and picture buttons) don't have their own labels, but you can position a text control above or beside these controls to serve as a de facto label for them

3.16 Assigning Accelerator Keys

Accelerator keys enable users to access dialog box controls simply by pressing Alt + a specified letter. Users can employ accelerator keys to choose a push button or an option button; toggle a check box on or off; and move the insertion point into a text box or group box or to the currently selected item in a list box, combo box, or drop list box.

An accelerator key is essentially a single letter that you designate for this purpose from a control's label. You can assign an accelerator key directly to controls that have their own label (option buttons, push buttons, check boxes, and group boxes). (You can't assign an accelerator key to OK and Cancel buttons because, as noted above, their labels can't be edited.) You can create a de facto accelerator key for certain controls that don't have their own labels (text boxes, list boxes, combo boxes, and drop list boxes) by assigning an accelerator key to an associated text control.

To assign an accelerator key

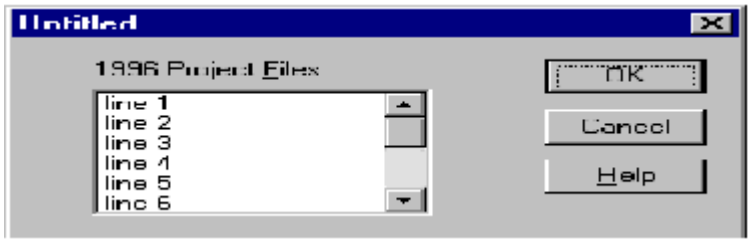
1. Here's how to designate a letter from a control's label to serve as the accelerator key for that control.
2. Display the Information dialog box for the control to which you want to assign an accelerator key.
3. In the Text$ field, type an ampersand (&) before the letter you want to designate as the accelerator key.
4. Click the OK button or press Enter.

   The letter you designated is now underlined on the control's label, and users will be able to access the control by pressing Alt + the underlined letter.

> **Attention**
> Accelerator key assignments must be unique within a particular dialog box. If you attempt to assign the same accelerator key to more than one control, Dialog Editor displays a reminder that letter has already been assigned.

5. If, for example, you have a push button whose label reads *Apply,* you can designate *A* as the accelerator key by displaying the Push Button Information dialog box and typing *&Apply* in the Text$ field. When you press Enter, the button label looks like the following illustration, and users will be able to choose the button by typing Alt+A.

6. As another example, let's say you have a list box that is immediately preceded in the dialog box template by a text control whose label reads *1994 Project Files.* By using the method described above, you can designate *F* as the accelerator key. When you click OK or press Enter, the text control label looks like the following illustration, and users will be able to move the insertion point to the currently selected item in the list box by typing Alt+F.



Tip

In order for such a de facto accelerator key to work properly, the text control or group box label to which you assign the accelerator key must be associated with the control(s) to which you want to provide user access-that is, in the dialog box template, the description of the text control or group box must immediately precede the description of the control(s) that you want associated with it. The simplest way to establish such an association is to create the text control or group box first, followed immediately by the associated control(s).

3.17 Specifying Pictures

In the preceding section, you learned how to add picture controls and picture button controls to your dialog box. But these controls are nothing more than empty outlines until you specify the pictures that you want them to display.

A picture control or picture button control can display a Windows bitmap or metafile, which you can obtain from a file or from a specified library. (Refer to the following subsection for information on creating or modifying picture libraries under Windows.).

To specify a picture from a file

1. Here's how to display a Windows bitmap or metafile from a file on a picture control or picture button control by using the control's Information dialog box to indicate the file in which the picture is contained.
2. Display the Information dialog box for the picture control or picture button control whose picture you want to specify.
3. In the Picture source option button group, select File.
4. In the Name$ field, enter the name of the file containing the picture you want to display in the picture control or picture button control.

   Note: By clicking the Browse button, you can display the Select a Picture File dialog box and use it to find the file.
5. Click the OK button or press Enter.

   The picture control or picture button control now displays the picture you specified.

To specify a picture from a picture library

1. Here's how to display a Windows bitmap or metafile from a library on a picture control or picture button control by first using the Dialog Box Information dialog box to specify the library and then using the control's Information dialog box to indicate the name of the picture.
2. Display the Dialog Box Information dialog box.
3. In the Picture Library field, specify the name of the picture library that contains the picture(s) you want to display in your dialog box.

   Note: By clicking the Browse button, you can display the Select a Picture Library dialog box and use it to find the library.

   If you specify a picture library in the Dialog Box Information dialog box, all the pictures in your dialog box must come from this library.
4. Click the OK button or press Enter.
5. Display the Information dialog box for the picture control or picture button control whose picture you want to specify.
6. In the Picture source option button group, select Library.
7. In the Name$ field, enter the name of the picture you want to display on the picture control or picture button control. (This picture must be from the library that you specified in step 2.).
8. Click the OK button or press Enter.

   The picture control or picture button control now displays the picture you specified.

3.18 Creating or Modifying Picture Libraries under Windows

The Picture statement in BasicScript allows images to be specified as individual picture files or as members of a picture library, which is a DLL that contains a collection of pictures. Currently, both Windows bitmaps and metafiles are supported. You can obtain a picture library either by creating a new one or by modifying an existing one, as described below.

Each image is placed into the DLL as a resource identified by its unique resource identifier. This identifier is the name used in the Picture statement to specify the image.

The following resource types are supported in picture libraries:

Resource Type	Description
2	Bitmap. This is defined in windows.h as RT_BITMAP.
256	Metafile. Since there is no resource type for metafiles, 256 is used.

To create a picture library under Windows

1. Here's how to create a new picture library to contain the Windows bitmaps or metafiles that you want to display on picture controls or picture button controls in your dialog box.
2. Create a C file containing the minimal code required to establish a DLL. The following code can be used:
   ```
   #include <;windows.h>
   int
   CALLBACK LibMain(
   HINSTANCE hInstance,
   WORD wDataSeg,
   WORD wHeapSz,
   LPSTR lpCmdLine) {
   UnlockData(0);
   return 1;}
   ```
3. Use the following code to create a DEF file for your picture library:
   ```
   LIBRARY
   DESCRIPTION “My Picture Library”
   EXETYPE WINDOWS
   CODE LOADONCALL MOVABLE DISCARDABLE
   DATA PRELOAD MOVABLE SINGLE
   HEAPSIZE 1024
   ```
4. Create a resource file containing your images. The following example shows a resource file using a bitmap called sample.bmp and a metafile called usa.wmf.
   ```
   #define METAFILE 256
   USA METAFILE “usa.wmf”
   MySample BITMAP “sample.bmp”
   ```
5. Create a make file that compiles your C module, creates the resource file, and links everything together.

To modify an existing picture library

1. Here's how to modify an existing picture library to contain the Windows bitmaps or metafiles that you want to display on picture controls or picture button controls in your dialog box.

2 Make a copy of the picture library you want to modify.

3 Modify the copy by adding images using a resource editor such as Borland's Resource Workshop or Microsoft's App Studio.

Tip

When you use a resource editor, you need to create a new resource type for metafiles (with the value 256).

3.19 Duplicating and Deleting Controls

To duplicate a control

1. Here's how to use Dialog Editor's duplicating feature, which saves you the work of creating additional controls individually if you need one or more copies of a particular control.
2. Select the control that you want to duplicate
3. Press Ctrl+D
   A duplicate copy of the selected control appears in your dialog box
4. Repeat step 2 as many times as necessary to create the desired number of duplicate controls
5. Duplicating is a particularly efficient approach if you need to create a group of controls, such as a series of option buttons or check boxes. Simply create the first control in the group and then, while the newly created control remains selected, repeatedly press Ctrl+D until you have created the necessary number of copies
6. Dialog Editor also enables you to delete single controls or even clear the entire dialog box.

To delete a single control

1. Select the control you want to delete.
2. Press Del.
   The selected control is removed from your dialog box.

To delete all the controls in a dialog box

1. Select the dialog box.
2. Press Del.
3. If the dialog box contains more than one control, Dialog Editor prompts you to confirm that you want to delete all controls. Click the Yes button or press Enter.
   All the controls disappear, but the dialog box's title bar and close box (if displayed) remain unchanged.

3.19.1 Undoing Editing Operations

You can undo editing operations that produce a change in your dialog box, including:

- The addition of a control
- The insertion of one or more controls from the Clipboard
- The deletion of a control
- Changes made to a control or dialog box, either with the mouse or with the Information dialog box

You cannot undo operations that don't produce any change in your dialog box, such as selecting controls or dialog boxes and copying material to the Clipboard.

To undo an editing operation:

- Press Ctrl+Z.
  Your dialog box is restored to the way it was before you performed the editing operation.

3.20 Editing an Existing Dialog Box

There are three ways to edit an existing dialog box: (1) You can copy the BasicScript template of the dialog box you want to edit from a script to the Clipboard and paste it into Dialog Editor. (2) You can use the capture feature to “grab” an existing dialog box from another application and insert a copy of it into Dialog Editor. (3) You can open a dialog box template file that has been saved on a disk. Once you have the dialog box displayed in Dialog Editor's application window, you can edit it using the methods described earlier in the chapter.

3.21 Pasting an Existing Dialog Box into Dialog Editor

You can use Dialog Editor to modify the BasicScript statements in your script that correspond to an entire dialog box or to one or more dialog box controls. To modify a BasicScript dialog box template contained in your script, you need to select the template and paste it into Dialog Editor for editing

To paste an existing dialog box into Dialog Editor

1. Copy the entire BasicScript dialog box template (from the Begin Dialog instruction to the End Dialog instruction) from your script to the Clipboard.
2. Open Dialog Editor.
3. Press Ctrl+V.
4. When Dialog Editor asks whether you want to replace the existing dialog box, click the Yes button.
   Dialog Editor creates a new dialog box corresponding to the template contained on the Clipboard.

To paste one or more controls from an existing dialog box into Dialog Editor

1. Copy the BasicScript description of the control(s) from your script to the Clipboard
2. Open Dialog Editor
3. Press Ctrl+V
   Dialog Editor adds to your current dialog box one or more controls corresponding to the description contained on the Clipboard

> **Attention**
> When you paste a dialog box template into Dialog Editor, the tabbing order of the controls is determined by the order in which the controls are described in the template. When you paste one or more controls into Dialog Editor, they will come last in the tabbing order, following the controls that are already present in the current dialog box.
>
> If there are any errors in the BasicScript statements that describe the dialog box or controls, the Dialog Translation Errors dialog box will appear when you attempt to paste these statements into Dialog Editor. This dialog box shows the lines of code containing the errors and provides a brief description of the nature of each error.

3.22 Capturing a Dialog Box

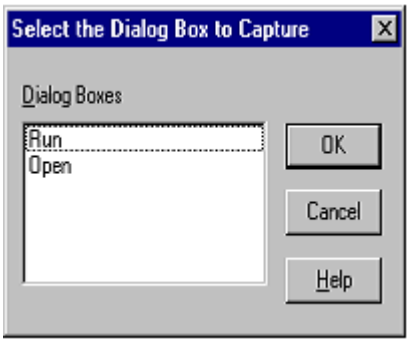
Here's how to capture the standard Windows controls from any standard Windows dialog box in another application and insert those controls into Dialog Editor for editing.

To capture an existing standard Windows dialog box

1 Display the dialog box you want to capture.

2 Open Dialog Editor.

3 Choose the Capture Dialog command from the File menu.
Dialog Editor displays a dialog box that lists all open dialog boxes that it is able to capture.



4 Select the dialog box you want to capture, and then click OK.
Dialog Editor now displays the standard Windows controls from the target dialog box.

Attention
Dialog Editor only supports standard Windows controls and standard Windows dialog boxes. Therefore, if the target dialog box contains both standard Windows controls and custom controls, only the standard Windows controls will appear in Dialog Editor's application window. If the target dialog box is not a standard Windows dialog box, you will be unable to capture the dialog box or any of its controls

3.23 Opening a Dialog Box Template File

Here's how to open any dialog box template file that has been saved on a disk so you can edit the template in Dialog Editor.

To open a dialog box template file

1. Choose Open from the File menu.
   The Open Dialog File dialog box appears.
2. Select the file containing the dialog box template that you want to edit and click the OK button.
   Dialog Editor creates a dialog box from the statements in the template and displays it in the application window.

> **Attention**
> If there are any errors in the BasicScript statements that describe the dialog box, the Dialog Translation Errors dialog box will appear when you attempt to load the file into Dialog Editor. This dialog box shows the lines of code containing the errors and provides a brief description of the nature of each error.

3.24 Testing an Edited Dialog Box

Dialog Editor lets you run your edited dialog box for testing purposes. When you click the Test tool, your dialog box comes alive, which gives you an opportunity to make sure it functions properly and fix any problems before you incorporate the dialog box template into your script.

Before you run your dialog box, take a moment to look it over for basic problems such as the following:

- Does the dialog box contain a command button—that is a default OK or Cancel button, a push button, or a picture button?
- Does the dialog box contain all the necessary push buttons?
- Does the dialog box contain a Help button if one is needed?
- Are the controls aligned and sized properly?
- If there is a text control, is its font set properly?
- Are the close box and title bar displayed (or hidden) as you intended?
- Are the control labels and dialog box title spelled and capitalized correctly?
- Do all the controls fit within the borders of the dialog box?
- Could you improve the design of the dialog box by adding one or more group boxes to set off groups of related controls?
- Could you clarify the purpose of any unlabeled control (such as a text box, list box, combo box, drop list box, picture, or picture button) by adding a text control to serve as a de facto label for it?
- Have you made all the necessary accelerator key assignments?

After you've fixed any elementary problems, you're ready to run your dialog box so you can check for problems that don't become apparent until a dialog box is activated.

Testing your dialog box is an iterative process that involves running the dialog box to see how well it works, identifying problems, stopping the test and fixing those problems, then running the dialog box again to make sure the problems are fixed and to identify any additional problems, and so forth—until the dialog box functions the way you intend.

To test your dialog box

1. Click the Run tool on the toolbar

   Or

   Press F5

   The dialog box becomes operational, and you can check how it functions.
2. To stop the test, click the Run tool, press F5, or double-click the dialog box's close box (if it has one).
3. Make any necessary adjustments to the dialog box.
4. Repeat steps 1-3 as many times as you need in order to get the dialog box working properly.

   When testing a dialog box, you can check for operational problems such as the following:

Tabbing order

1. When you press the Tab key, does the focus move through the controls in a logical order?(Remember, the focus skips over items that users cannot act upon, including group boxes, text controls, and pictures.)
2. When you paste controls into your dialog box, Dialog Editor places their descriptions at the end of your dialog box template, in the order in which you paste them in. Therefore, you can use a simple cut-and-paste technique to adjust the tabbing order. First, click the Run tool to end the test and then, proceeding in the order in which you want the controls to receive the focus, select each control, cut it from the dialog box (by pressing Ctrl+X), and immediately paste it back in again (by pressing Ctrl+V). The controls will now appear in the desired order in your template and will receive the tabbing focus in that order.

Option button grouping

1. Are the option buttons grouped correctly? Does selecting an unselected button in a group automatically deselect the previously selected button in that group?
2. To merge two groups of option buttons into a single group, click the Run tool to end the test and then use the Option Button Information dialog box to assign the same. Option Group name for all the buttons that you want included in that group.

Text box functioning

1. Can you enter only a single line of nonwrapping text, or can you enter multiple lines of wrapping text?
2. If the text box doesn't behave the way you intended, click the Run tool to end the test; then display the Text Box Information dialog box and select or clear the Multiline check box.

Accelerator keys

1. If you have assigned an accelerator key to a text control or group box in order to provide user access to a text box, list box, combo box, drop list box, or group box, do the accelerator keys work properly? That is, if you press Alt + the designated accelerator **key**, does the insertion point move into the text box or group box or to the currently selected item in the list box, combo box, or drop list box?
2. If the accelerator key doesn't work properly, it means that the text box, list box, combo box, drop list box, or group box is not associated with the text control or group box to which you assigned the accelerator key—that is, in your dialog box template, the description of the text control or group box does not immediately precede the description of the control(s) that should be associated with it. As with tabbing-order problems (discussed above), you can fix this problem by using a simple cut-and-paste technique to adjust the order of the control descriptions in your template. First, click the Run tool to end the test; then cut the text control or group box from the dialog box and immediately paste it back in again; and finally, do the same with each of the controls that should be associated with the text control or group box. The controls will now appear in the desired order in your template, and the accelerator keys will work properly.

3.25 Incorporating a Dialog Box into Your Script

Dialog boxes and dialog box controls are communicated between Dialog Editor and your script via the Clipboard, where they are represented as BasicScript statements. Here's how to copy a dialog box or control and paste it into your script.

To incorporate a dialog box or control into your script

1. Select the dialog box or control that you want to incorporate into your script.
2. Press Ctrl+C.
3. Open your script and paste in the contents of the Clipboard at the desired point.
   The dialog box template or control is now described in BasicScript statements in your script.

3.26 Exiting from Dialog Editor

Here's how to get out of Dialog Editor. When you exit, you can save your dialog box template (that is, the BasicScript description of the dialog box) as a text file.

To exit from Dialog Editor

1. Press Alt+F4.

   If you have made changes to your dialog box template, Dialog Editor asks whether you want to save those changes.

2. If you want to save your changes to a text file, click the Yes button.

   Dialog Editor displays the Save Dialog File dialog box, which you can use to specify the file to which you want to save your template.

3.27 Using a Custom Dialog Box in Your Script

After using Dialog Editor to insert a custom dialog box template into your script, you'll need to make the following modifications to your script.

1. Create a *dialog record* by using a Dim statement.
2. Put information into the dialog box by assigning values to its controls.
3. Display the dialog box by using either the Dialog() function or the Dialog statement.
4. Retrieve values from the dialog box after the user closes it.

3.28 Creating a Dialog Record

To store the values retrieved from a custom dialog box, you create a dialog record with a Dim statement, using the following syntax:

Dim DialogRecord As DialogVariable

Here are some examples of how to create dialog records:

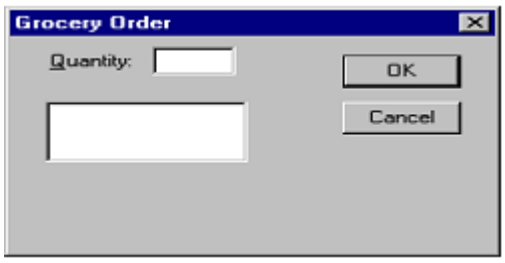
```
Dim b As UserDialog 'Define a dialog record “b”.
Dim PlayCD As CDDialog 'Define a dialog record
'“PlayCD”.
```

Here is a sample script, which can be found on your disk under the name dialog1.ebs, that illustrates how to create a dialog record named b within a dialog box template named UserDialog. Notice that the order of the statements within the script is as follows: the dialog box template precedes the statement that creates the dialog record, and the Dialog statement follows both of them.

```
Sub Main()
Dim ListBox1$() 'Initialize list box array.
'Define the dialog box template.
Begin Dialog UserDialog ,,163,94,“Grocery Order”
Text 13,6,32,8,“&Quantity:”,.Text1
TextBox 48,4,28,12,.TextBox1
ListBox 12,28,68,32,ListBox1$,.ListBox1
OKButton 112,8,40,14
CancelButton 112,28,40,14
End Dialog
Dim b As UserDialog 'Create the dialog record.
Dialog b 'Display the dialog box.
End Sub
```

3.29 Putting Information into the Dialog Box

If you open and run the sample script shown in the preceding subsection, you'll see a dialog box that resembles the following (this dialog box is from BasicScript for Win32 running under Windows 95, so the dialog box you see may be slightly different).



As you can see, this custom dialog box isn't very useful. For one thing, the user doesn't see any items in the list box along the left side of the dialog box. To put information into this dialog box, you assign values to its controls by modifying the statements in your script that are responsible for displaying those controls to the user. The following table lists the dialog box controls to which you can assign values and the types of information you can control.

Control(s)	Types of Information
List box, drop list box, and combo box.	Items
Text box.	Default text
Check box	Values

In the following subsections, you'll learn how to define and fill an array, set the default text in a text box, and set the initial focus and tab order for the controls in your custom dialog box.

3.29.1 Defining and Filling an Array

You can store items in the list box shown in the example above by creating an array and then assigning values to the elements of the array. For example, you could include the following lines to initialize an array with three elements and assign the names of three common fruits to these elements of your array:

```
Dim ListBox1$(3) 'Initialize list box array.
ListBox1$(0) = “Apples”
ListBox1$(1) = “Oranges”
ListBox1$(2) = “Pears”
```

3.29.2 Setting Default Text in a Text Box

You can set the default value of the text box in your script to 12 with the following statement, which must follow the statement that defines the dialog record but precede the statement or function that displays the custom dialog box:

```
b.TextBox1 = “12”
```

3.29.3 Setting the Initial Focus and Controlling the Tabbing Order

You can determine which control has the focus when your custom dialog box is first displayed as well as the tabbing order between controls by understanding two rules that BasicScript follows. First, the focus in a custom dialog box is always set initially to the first control to appear in the dialog box template. Second, the order in which subsequent controls appear within the dialog box template determines the tabbing order. That is, pressing

the Tab key will change the focus from the first control to the second one, pressing the Tab key again will change the focus to the third control, and so on.

3.30 Displaying the Custom Dialog Box

To display a custom dialog box, you can use either a Dialog() function or a Dialog statement.

3.30.1 Using the Dialog() Function

You can use a Dialog() function to determine how the user closed your custom dialog box. For example, the following statement will return a value when the user clicks an OK button or a Cancel button or takes another action:

response% = Dialog(b)

The Dialog() function returns any of the following values:

Value Returned	If
-1	The user clicked the OK button.
0	The user clicked the Cancel button.
>0	The user clicked a push button. The returned number represents which button was clicked based on its order in the dialog box template (1 is the first push button, 2 is the second push button, and so on).

3.30.2 Using the Dialog Statement

You can use the Dialog statement when you don't need to determine how the user closed your dialog box. You'll still be able to retrieve other information from the dialog box record, such as the value of a list box or other dialog box control. The following is an example of the correct use of the Dialog statement:

Dialog b

3.31 Retrieving Values from the Custom Dialog Box

After displaying a custom dialog box for your user, your script must retrieve the values of the dialog controls. You retrieve these values by referencing the appropriate identifiers in the dialog record.

You'll find an example of how to retrieve values from a custom dialog box in the following subsection.

Sample

In the following sample script, a version of which can be found on your disk under the name dialog2.ebs, several of the techniques described earlier in this section have been used.

In this script, the array named ListBox1 is filled with three elements (“Apples”, “Oranges”, and “Pears”). The default value of TextBox1 is set to 12. A variable named response is used to store information about how the custom dialog box was closed. An identifier named ListBox1 is used to determine whether the user chose “Apples”, “Oranges”, or “Pears” in the list box named ListBox$. Finally, a Select Case…End Select statement is used to display a message box appropriate to the manner in which the user dismissed the dialog box.

```
Sub Main()
Dim ListBox1$(2) 'Initialize list box array.
Dim response%
ListBox1$(0) = “Apples”
ListBox1$(1) = “Oranges”
ListBox1$(2) = “Pears”
Begin Dialog UserDialog ,,163,94,“Grocery Order”
Text 13,6,32,8,“&Quantity:”,.Text1 'First control
'in the
'template gets
'focus.
TextBox 48,4,28,12,.TextBox1
ListBox 12,28,68,32,ListBox1$,.ListBox1
 OKButton 112,8,40,14
CancelButton 112,28,40,14
End Dialog
Dim b As UserDialog 'Create the dialog record.
b.TextBox1 = “12” 'Set default value of the
'text box to 1 dozen.
response = Dialog(b) 'Display the dialog box.
Select Case response%
Case -1
Fruit$ = ListBox1$(b.ListBox1)
MsgBox “Thank you for ordering ” + _
b.TextBox1 + “ ” + Fruit$ + “.”
Case 0
MsgBox “Your order has been canceled.”
End Select
End Sub
```

3.32 Using a Dynamic Dialog Box in Your Script

The preceding section explained how you can use a custom dialog box in your script. As you learned, you can retrieve the values from dialog box controls after the user dismisses the dialog box by referencing the identifiers in the dialog record.

You can also retrieve values from a custom dialog box while the dialog box is displayed, using a feature of BasicScript called *dynamic dialog boxes.*

The following script, named dialog3.ebs on your disk, illustrates the most important concepts you'll need to understand in order to create a dynamic dialog box in your script:

```
'Dim “Fruits” and “Vegetables” arrays here to make them
'accessible to all procedures.
Dim Fruits(2) As String
Dim Vegetables(2) As String
'Dialog procedure--must precede the procedure that defines 'the custom dialog box.
Function DialogControl(ctrl$, action%, suppvalue%) As Integer
Select Case action%
Case 1
DlgListBoxArray “ListBox1”, fruits 'Fill list box
'with items
'before dialog 'box is visible.
DlgValue “ListBox1”, 0 'Set default value to
'first item in list box.
Case 2
'Fill the list box with names of fruits or
'vegetables when the user selects an option button.
If ctrl$ = “OptionButton1” Then
DlgListBoxArray “ListBox1”, fruits
DlgValue “ListBox1”, 0
ElseIf ctrl$ = “OptionButton2” Then
DlgListBoxArray “ListBox1”, vegetables
DlgValue “ListBox1”, 0
End If
End Select
End Function
Sub Main()
Dim ListBox1$() 'Initialize array for use by
'ListBox statement in template.
Dim Produce$
'Assign values to elements in the “Fruits” and
'“Vegetables” arrays.
Fruits(0) = “Apples”
Fruits(1) = “Oranges”
Fruits(2) = “Pears”
Vegetables(0) = “Carrots”
Vegetables(1) = “Peas”
Vegetables(2) = “Lettuce”
'Define the dialog box template.
Begin Dialog UserDialog ,,163,94,“Grocery Order”, _
.DialogControl
Text 13,6,32,8,“&Quantity:”,.Text1 'First control
 'in template
'gets the focus.
TextBox 48,4,28,12,.TextBox1
ListBox 12,28,68,32,ListBox1$,.ListBox1
OptionGroup .OptionGroup1
 OptionButton 12,68,48,8,“&Fruit”,.OptionButton1
OptionButton 12,80,48,8,“&Vegetables”,.OptionButton2
OKButton 112,8,40,14
CancelButton 112,28,40,14
End Dialog
Dim b As UserDialog 'Create the dialog record.
b.TextBox1 = “12” 'Set the default value of the
'text box to 1 dozen.
response% = Dialog(b) 'Display the dialog box.
Select Case response%
Case -1
If b.OptionGroup1 = 0 Then
produce$ = fruits(b.ListBox1)
Else
produce$ = vegetables(b.ListBox1)
End If
MsgBox “Thank you for ordering ” & _
b.TextBox1 & “ “ & produce$ & ”.”
Case 0
```

```
        MsgBox “Your order has been canceled.”
        End Select
        End Sub
```

3.33 Making a Dialog Box Dynamic

The first thing to notice about the preceding script, which is a more complex variation of the dialog2.ebs script described earlier in this chapter, is that an identifier named *.DialogControl* has been added to the Begin Dialog statement. As you will learn in the following subsection, this parameter to the Begin Dialog statement tells BasicScript to pass control to a function procedure named DialogControl.

3.33.1 Using a Dialog Function

Before BasicScript displays a custom dialog box by executing a Dialog statement or Dialog() function, it must first initialize the dialog box. During this initialization process, BasicScript checks to see whether you've defined a dialog function as part of your dialog box template. If so, BasicScript will give control to your dialog function, allowing your script to carry out certain actions, such as hiding or disabling dialog box controls.

After completing its initialization process, BasicScript displays your custom dialog box. When the user selects an item in a list box, clears a check box, or carries out certain other actions within the dialog box, BasicScript will again call your dialog function.

In fact, BasicScript also calls your dialog function repeatedly even while the user is not interacting with the dialog box. You can use this fact to update a dialog box continuously, as with the stwatch.ebs sample on your disk.

3.33.2 Responding to User Actions

A BasicScript dialog function can respond to six types of user actions:

Action	Description
1	This action is sent immediately before the dialog box is shown for the first time.
2	This action is sent when: • A button is clicked, such as OK, Cancel, or a push button. • A check box's state has been modified. • An option button is selected. In this case, *ControlName$* contains the name of the option button that was clicked, and *SuppValue* contains the index of the option button within the option button group (0 is the first option button, 1 is the second, and so on). • The current selection is changed in a list box, drop list box, or combo box. In this case, *ControlName$* contains the name of the list box, combo box, or drop list box, and *SuppValue* contains the index of the new item (0 is the first item, 1 is the second, and so on).
3	This action is sent when the content of a text box or combo box has been changed *and* that control loses focus.
4	This action is sent when a control gains the focus.
5	This action is sent continuously when the dialog box is idle.
6	This action is sent when the dialog box is moved.

3.34 Menu/Tools Reference

File Menu

Menu Command	Keyboard Shortcut	Function
*N*ew	Ctrl+N	Creates a new dialog box. Dialog Editor prompts you before discarding any changes you have made to your current dialog box.
*O*pen...	Ctrl+O	Displays the Open Dialog File dialog box, which you can use to open an existing dialog box template. Dialog Editor prompts you before discarding any changes you have made to your current dialog box.
*S*ave	Ctrl+S	If you have already created a file for the current dialog box template, saves the template to that file. If you have not yet created a file for the current dialog box template, displays the Save Dialog File dialog box, which you can use to specify the file to which you want to save the current template.
Save *A*s...		Displays the Save Dialog File dialog box, which you can use to save the current dialog box template in a file under a new name.
*T*est Dialog	F5	Toggles between the run mode (in which the dialog box “comes alive” for testing purposes) and the edit mode (in which you can make changes to the dialog box).
*C*apture Dialog		Captures the standard Windows controls from a standard Windows dialog box in another Windows application.
Exit & Return	Alt+F4	Closes Dialog Editor and returns you to the host application. Dialog Editor prompts you before discarding any changes you have made to your current dialog box.

Edit Menu

Menu Command	Keyboard Shortcut	Function
*U*ndo	Ctrl+Z	Allows you to undo up to 10 preceding operations. The Undo command continually indicates the next operation you can undo by selecting it and grays out when there are no more operations that can be undone.
Cu*t*	Ctrl+X	Removes the selected dialog box or control from Dialog Editor's application window and places it on the Clipboard.
*C*opy	Ctrl+C	Copies the selected dialog box or control, without removing it from Dialog Editor's application window, and places it on the Clipboard.
*P*aste	Ctrl+V	Inserts the contents of the Clipboard into Dialog Editor. If the Clipboard contains BasicScript statements describing one or more controls, then those controls are added to the current dialog box. If the Clipboard contains the BasicScript template for an entire dialog box, then Dialog Editor creates a new dialog box from the statements in the template. If the BasicScript statements contain errors, Dialog Editor displays the Dialog Translation Errors dialog box, which helps you pinpoint the location and nature of the errors.
Clear	Del	Removes the selected dialog box or control from Dialog Editor's application window without placing it on the Clipboard. If you have selected the dialog box and it contains more than one control, Dialog Editor prompts you before removing all the controls from it.
Duplicate	Ctrl+D	Creates a duplicate copy of the selected control.

Menu Command	Keyboard Shortcut	Function
Size to Text	F2	Adjusts the borders of certain controls to fit the text displayed on them.
*I*nfo...	Ctrl+I	Displays the Information dialog box for the selected dialog box or control. You can use this dialog box to check and adjust various attributes of controls and dialog boxes.
*G*rid...	Ctrl+G	Displays the Grid dialog box, which you can use to display or turn off the grid and adjust the grid's spacing.

Controls Menu

Menu Command	Toolbar Tool	Function
O*K* button		Adds a default OK button to your dialog box. An OK button is a special type of push, or command, button.
C*a*ncel button		Adds a default Cancel button to your dialog box. A Cancel button is a special type of push, or command, button
Help button		Adds a Help button to your dialog box. A Help button is a special type of push, or command, button.
*P*ush button		Adds a push, or command, button to your dialog box.
*O*ption button		Adds an option button to your dialog box. An option button is one of a group of two or more linked buttons that let users select only one from a group of mutually exclusive choices.
*C*heck box		Adds a check box to your dialog box. Users can check or clear a check box to indicate their preference regarding the alternative specified on the check box label.
*G*roup box		Adds a group box to your dialog box. A group box is a rectangular design element used to enclose a group of related controls. You can use the optional group box label to display a title for the controls in the box.
*T*ext		Adds a text control to your dialog box. A text control is a field containing text that you want to display for the users' information. The text in this field wraps, and the field can contain a maximum of 255 characters. Text controls can either display stand-alone text or be used as labels for text boxes, list boxes, combo boxes, drop list boxes, pictures, and picture buttons. You can choose the font in which the text appears.
T*e*xt box		Adds a text box to your dialog box. A text box is a field into which users can enter text (potentially, as much as 32K). By default, this field holds a single line of nonwrapping text. If you choose the Multiline setting in the Text Box Information dialog box, this field will hold multiple lines of wrapping text.
*L*ist box		Adds a list box to your dialog box. A list box is a displayed, scrollable list from which users can select one item. The currently selected item is highlighted on the list.
Com*b*o box		Adds a combo box to your dialog box. A combo box consists of a text field with a displayed, scrollable list beneath it. Users can either select an item from the list or enter the name of the desired item in the text field. The currently selected item is displayed in the text field. If the item was selected from the scrolling list, it is highlighted there as well.

Menu Command	Toolbar Tool	Function
*D*rop list box		Adds a drop list box to your dialog box. A drop list box consists of a field that displays the currently selected item, followed by a downward-pointing arrow, which users can click to temporarily display a scrolling list of items. Once they select an item from the list, the list disappears and the newly selected item is displayed in the field.
P*i*cture		Adds a picture to your dialog box. A picture is a field used to display a Windows bitmap or metafile.
Picture button		Adds a picture button to your dialog box. A picture button is a special type of push, or command, button on which a Windows bitmap or metafile appears.

Help Menu

Menu Command	Keyboard Shortcut	Function
Contents	F1	Presents a list of major topics in the Help system. By clicking a topic on the list, you can display the available Help information about that topic.
Search for Help On...		Displays a dialog box that allows users to search for Help topics containing specific keywords.
About Dialog Editor...		Displays the About Dialog Editor dialog box, which indicates application name, version number, copyright statement, and icon, as well as additional information such as amount of available memory and disk space.

4 Recording Scripts

Script Recorder is a utility that you can use to record actions you perform with Microsoft Windows applications for later playback. Script Recorder watches the actions you perform and translates them into BasicScript code, which you can add to a BasicScript script. Once you have recorded a task using Script Recorder, you can perform the same task at a later time by playing back the script that was created. The following topics are discussed in this section.

- Script Recorder Basics
- Using Script Recorder
- Recording a Script
- Actions Recorded
- Examining and Adjusting Your Script's BasicScript Code

4.1 Script Recorder Basics

When you invoke Script Recorder, it watches your interaction with Windows applications and translates the keystrokes you press and the mouse actions you perform into a script consisting of BasicScript statements. Since recording occurs at the system level, the resulting script is not tied to any particular application and thus can include actions involving a number of applications.

When it is active, Script Recorder watches events as they are passed to various windows and records those events into an event buffer. If directed to do so, the Recorder removes and collapses certain events into high-level statements. (For example, a mouse action can be removed from the event buffer and replaced with a mouse click. Similarly, two mouse clicks can be removed from the event buffer and replaced with a mouse double click.).

After you stop recording, Script Recorder's event buffer is reduced via a series of steps, each of which results in the replacement of certain event sequences with a single higher-level event. After reduction in this manner, the event buffer is converted into BasicScript statements in the text buffer. Depending on the implementation, you may be able to save the resulting text buffer to a file or return it to the host application.

In the following sections, you'll learn the purpose of the controls on Script Recorder's control panel, how to record a script, and what actions the Recorder records and what BasicScript statements it outputs as a result. The chapter ends with some suggestions for adjustments you might want to make to the BasicScript code generated by Script Recorder.

4.2 Features of Script Recorder

Script Recorder supports the following features:

- The Recorder runs under Windows 3.1.
- The Recorder recognizes most standard Windows actions and converts them into BasicScript statements. (The “Actions Recorded” section of the chapter explains the types of Windows actions the Recorder recognizes and lists the BasicScript statements that it outputs for each type of action.)
- The number of events that you can record with Script Recorder is limited by the size of the Recorder's internal event buffer. (The size of the event buffer is 64K, which means you can record about 7,280 events.)
- The text buffer used to convert events into BasicScript code is also limited to 64K in size.
- Since Script Recorder does not record timing information, you can play your recorded script back at full speed regardless of any delays that may have occurred while you were recording it.

4.3 Using Script Recorder

Script Recorder's application interface consists of the following VCR-style three-button control panel:

- Record button: When you first invoke Script Recorder, you won't need to click the Record button because it will already be in a “pushed,” or disabled, state to indicate that the Recorder is now recording. If you pause the Recorder, clicking the Record button will resume your recording session.
- Pause button: If you need to take a break in your recording session, clicking the Pause button will suspend recording. Script Recorder won't record any actions you perform while the Pause button is pushed.
- End button: Clicking the End button will stop Script Recorder and end your recording session.

Script Recorder doesn't record anything you do to its control panel, so you can stop, start, or suspend recording as often as you wish and move the panel to different areas of your screen as you proceed with your recording session.

4.4 Recording a Script

Once Script Recorder has recorded a series of actions and translated them into BasicScript statements, you will probably want to examine the code in that script and make some adjustments and improvements, add comments, and so forth, as discussed later in the chapter.

However, if you do a bit of advance planning before you invoke Script Recorder, you can produce a script that will run better and require fewer adjustments. Simply keep in mind that Script Recorder records *all* the actions you take while the Recorder is running (including mistakes and unnecessarily cumbersome ways of doing things) and translates those actions into BasicScript code. You should therefore plan to carry out the tasks you are recording as expeditiously as possible.

Depending on the implementation, the following recording options may be available when you use Script Recorder.

- Literal recording: Prevents high-level event reduction. Only literal mouse and keyboard activity will be recorded.
- Mouse moves: Records mouse movement when the mouse button is pressed.
- Unpressed mouse moves: Records mouse movement when the mouse button is not pressed.
- Relative: Records mouse positions relative to the active window.
- Comments: Adds a comment at the end of each recorded BasicScript statement.
- Menu command: Records only a single menu command, then stops recording.
- Keyboard: Enables keyboard recording.
- Mouse: Enables mouse recording.

To record a script

1. Invoke Script Recorder.

   Script Recorder's three-button control panel appears.

   Note: The first time the control panel appears, you won't need to click the Record button because it will already be in a “pushed,” or disabled, state to indicate that the Recorder is now recording.
2. Perform the series of events that you want to record, clicking the Pause and Record buttons to suspend and restart the recording session as often as necessary.
3. To stop the recording session, click the End button.

   Depending on the implementation, when you finish recording, you may be able to save your BasicScript script to a file, copy it to the Clipboard, or return it to the host application.

4.5 Actions Recorded

Script Recorder recognizes the following types of Windows actions:

- Application focus switch
- Window scrolling
- Mouse activity
- Keyboard activity
- Window management
- Menu commands
- Dialog box interaction

4.6 Application Focus Switch

Script Recorder recognizes when an application receives the focus and records this as an AppActivate statement. The focus can be shifted by using any of the standard Windows application-switching methods, such as pressing Alt+Tab or Ctrl+Esc, clicking the mouse on another application, or selecting the Switch To command from an application's Control menu.

Script Recorder also records window focus shifts to ensure that subsequent actions occur (and are synchronized) within the appropriate window.

Statements	
AppActivat	WinActivat

4.7 Window Scrolling

Script Recorder records interactions with windows that respond to scrolling messages (for example, windows that have built-in Windows scroll bars, such as Notepad and Write).

Statements	
VLine	HLine
VPage	HPage
VScroll	HScroll

4.8 Mouse Activity

Script Recorder compresses mouse down/up actions into high-level forms, and it completely compresses out mouse activity that leads to a recognizable result, such as resizing a window.

Statements	
QueMouseClick	QueMouseMove
QueMouseDblClk	QueMouseDblDn
QueMouseDn	QueMouseUp
QueSetRelativeWindow	QueFlush

4.9 Keyboard Activity

Script Recorder converts keyboard press/release actions, and it converts complex key combinations, such as Ctrl +Esc or Alt+Shift+C, into a readable form following Visual Basic SendKeys () syntax. It completely compresses out keyboard activity that results in a higher-level command, such as the following keystroke sequence, which results in a window reposition: Alt+Space, M, Right arrow, Right arrow, Right arrow, Enter.

Script Recorder also records keyboard activity that modifies mouse events, such as pressing Shift while clicking the left mouse button.

Statements	
QueKeyDn	QueKeyUp
QueKeys	DoKeys
SendKeys()	QueFlush

4.10 Window Management

Script Recorder recognizes high-level interactions with windows, such as moving, sizing, activating, minimizing, maximizing, and restoring. The Recorder differentiates between interactions with applications, pop-up windows, and child MDI windows.

Statements	
AppMaximize	WinMaximize
AppMinimize	WinMinimize
AppMove	WinMove
AppSize	WinSize
AppRestore	WinRestore

4.11 Menu Commands

Script Recorder watches for interactions with an application's built-in menus and records the results of those interactions. It does not record any intermediate keyboard and mouse events used to select a menu command.

Statement	
Menu	

4.12 Dialog Box Interaction

Script Recorder recognizes interactions with standard Windows controls (that is, text boxes, push buttons, check boxes, option buttons, list boxes, combo boxes, and drop list boxes). It also recognizes interactions with standard controls of certain other applications, such as Borland's custom controls and the controls in Visual Basic. When recording interactions with dialog boxes containing standard controls, Script Recorder records the results of interactions with the controls rather than the mouse/keyboard actions that produced those results.

The Recorder also recognizes mixed dialog boxes—that is, dialog boxes containing both standard controls and custom controls. Many applications-such as Control Panel, Corel Draw, and Ami Pro—employ mixed dialog boxes. When recording interactions with mixed dialog boxes, Script Recorder records interactions with standard controls normally (that is, it records the results of those interactions), and it records interactions with custom controls using high—level mouse and keyboard statements.

Statements	
ActivateControl	SetCheckBox
SelectComboBoxItem	SetEditText
SelectButton	SetOption
SelectListBoxItem	

4.13 Examining and Adjusting Your Script's BasicScript Code

Once you have used Script Recorder to record a series of actions and translate them into a BasicScript script, you could simply save the script as is for future use. However, in order to ensure that your script functions optimally, you'll probably want to examine the BasicScript statements generated by Script Recorder and make the following types of minor adjustments:

- You should make sure that the events recorded by Script Recorder are those that you really intended to be in your script and that the script does what you want it to do when it is played back.
- You may want to make improvements to your script.
- If you plan to incorporate the recording into a script you are creating, you will need to integrate it into that script.
- If Script Recorder has trouble recording a nonstandard action, it may record a series of mouse and keyboard events rather than reducing them into a single higher-level event. You should therefore check your code to make sure it contains no extraneous mouse or keyboard events.
- When your script is played back, errors may be generated if the system's configuration differs from the conditions under which you recorded the script. You'll therefore need to build in provisions to enable the script to respond to such errors.
- You may want to add comments to your code right after you record the script, while your rationale for doing things a certain way is still fresh in your mind.
- Since Script Recorder does not record timing information, your script normally plays back at full speed. If you do not want it to do so, you can intersperse artificial delays into the recorded output to slow the playback of the statements.

5 Notices

Trademarks

Experion®, PlantScape®, SafeBrowse®, TotalPlant®, and TDC 3000® are registered trademarks of Honeywell International, Inc.

OneWireless™ is a trademark of Honeywell International, Inc.

Other trademarks

Microsoft and SQL Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Trademarks that appear in this document are used only to the benefit of the trademark owner, with no intention of trademark infringement.

Third-party licenses

This product may contain or be derived from materials, including software, of third parties. The third party materials may be subject to licenses, notices, restrictions and obligations imposed by the licensor. The licenses, notices, restrictions and obligations, if any, may be found in the materials accompanying the product, in the documents or files accompanying such third party materials, in a file named third_party_licenses on the media containing the product, or at http://www.honeywell.com/ps/thirdpartylicenses.

5.1 Documentation feedback

You can find the most up-to-date documents on the Honeywell Process Solutions support website at:

http://www.honeywellprocess.com/support

If you have comments about Honeywell Process Solutions documentation, send your feedback to:

hpsdocs@honeywell.com

Use this email address to provide feedback, or to report errors and omissions in the documentation. For immediate help with a technical problem, contact your local Honeywell Process Solutions Customer Contact Center (CCC) or Honeywell Technical Assistance Center (TAC) listed in the “Support and other contacts” section of this document.

5.2 How to report a security vulnerability

For the purpose of submission, a security vulnerability is defined as a software defect or weakness that can be exploited to reduce the operational or security capabilities of the software.

Honeywell investigates all reports of security vulnerabilities affecting Honeywell products and services.

To report a potential security vulnerability against any Honeywell product, please follow the instructions at:

https://honeywell.com/pages/vulnerabilityreporting.aspx

Submit the requested information to Honeywell using one of the following methods:

- Send an email to security@honeywell.com.

  or

- Contact your local Honeywell Process Solutions Customer Contact Center (CCC) or Honeywell Technical Assistance Center (TAC) listed in the “Support and other contacts” section of this document.

5.3 Support

For support, contact your local Honeywell Process Solutions Customer Contact Center (CCC). To find your local CCC visit the website, https://www.honeywellprocess.com/en-US/contact-us/customer-support-contacts/Pages/default.aspx.

5.4 Training classes

Honeywell holds technical training classes on Experion PKS. These classes are taught by experts in the field of process control systems. For more information about these classes, contact your Honeywell representative, or see http://www.automationcollege.com.