Power Toolbox help index

For information on Power Toolbox, choose one of the categories below. To learn how to use help, press F1 or choose Using Help from the Help menu.

Using Power Toolbox with a keyboard
Commands
Dialog boxes
hDC Command Builder
Enhanced Command syntax

Commands

To see a description of a command, select the appropriate topic below.

File menu commands
New
Open...
Save
Save As...
Exit

Edit menu commands
Undo
Cut
Copy
Paste
Clear
Paste Special...
Delete.../Delete Row/Delete Column
Insert.../Insert Paste.../Insert Row/Insert Column
Button Size...
Select All

Toolbox menu commands
New...
Copy...
Rename...
Delete...
Application Associations...

Help menu commands
Index
Keyboard
Commands
Using Help
About...

Dialog boxes

To see a description of the Power Toolbox main window or of a dialog box, select the appropriate topic below.

Power Toolbox window

Application Associations dialog box
Button Size dialog box
Copy Toolbox dialog box
Delete Toolbox dialog box
New Toolbox dialog box
Rename Toolbox dialog box
Stretch Bitmap dialog box

New command

Opens a new, untitled toolbox file. If the open file contains unsaved changes, a prompt appears, asking if you want to save those changes before loading the new file.

Related topics

Open... command

Open... command

Opens a Power Toolbox file. Power Toolbox files have a .TBL file extension.

Save command

Saves changes to the open toolbox file, displaying the File Save As dialog box if the file has not yet been named.

Save As... command

Saves a new file, or saves an existing file under a new name or in a different location.

Exit command

Closes the Power Toolbox application. If changes have been made to the open file, or if the file is new and hasn't been saved, a prompt appears asking if you want to save those changes before exiting.

Related topics

Save command

Undo command

Reverses the last action (providing that action can be reversed).

Cut command

Removes the bitmaps and Enhanced Commands for the buttons selected in the toolbox and places the bitmaps and Enhanced Commands on the Clipboard. Using the Cut command is equivalent to clicking the Cut button.

Related topics

Paste command
Power Toolbox window

Copy command

Copies the bitmaps and Enhanced Commands for the buttons selected in the toolbox graphic and places the bitmaps and Enhanced Commands on the Clipboard. Using the Copy command is equivalent to clicking the Copy button.

Related topics

Paste command
Power Toolbox window

Paste command

Pastes the bitmaps and Enhanced Command assignments stored in the Clipboard, inserting the Enhanced Command (if any) in the **Command:** box and inserting the bitmaps (if any) onto the buttons selected in the toolbox graphic. Using the Paste command is equivalent to clicking the Paste button.

To paste only a bitmap or Enhanced Command when both are stored on the Clipboard, use the Paste Special... command.

You can copy a button-sized portion of the screen to the Clipboard with the capture tool.

Related topics

Copy command
Cut command
Paste Special... command
Power Toolbox window

Clear command

Deletes the contents of the buttons selected in the toolbox. Using the Clear command is equivalent to clicking the Clear button.

Related topics

Copy command
Cut command
Power Toolbox window

Paste Special... command

Lets you specify whether you want to paste the bitmap or the Enhanced Command stored on the Clipboard. If you want to paste both, you can simply use the Paste command.

You can copy a button-sized portion of the screen to the Clipboard with the capture tool.

Related topics
Power Toolbox window

Delete.../Delete Row/Delete Column command

The Delete... command is available when you select less than a full row or column of buttons. The Delete... command displays a dialog box in which you can specify how remaining buttons will move after the deletion.

When you select an entire row of buttons, the command becomes Delete Row. When you select an entire column of buttons, the command becomes Delete Column. These commands delete the selected buttons without displaying a dialog box.

Related topics

Power Toolbox window

Insert.../Insert Paste.../Insert Row/Insert Column command

The Insert... command is available when you select less than a full row or column of buttons. The Insert... command displays a dialog box in which you can specify how existing buttons will move to accommodate the insertion. If the number of selected buttons does not match the number of buttons on the Clipboard, Power Toolbox displays an error message.

The Insert Paste... command is available when the Clipboard contains button bitmaps and commands. The Insert Paste dialog box allows you to choose whether to paste only the bitmaps or commands, or both.

When you select an entire row of buttons, the command becomes Insert Row. When you select an entire column of buttons, the command becomes Insert Column. These commands insert a row or column without displaying a dialog box.

Related topics

Power Toolbox window

Button Size... command

The Button Size... command lets you change the size of the toolbox buttons and the capture tool.

Related topics

Button Size dialog box
Power Toolbox window

Button Size dialog box

Use the Button Size dialog box to change the size of the toolbox buttons and the capture tool.

Stretch Bitmap: list Select a direction to specify whether you want the button bitmaps to stretch horizontally, vertically, both, or not at all.

Button Width: Displays the width of one toolbox button in pixels.

Button Height: Displays the height of one toolbox button in pixels.

Toolbox graphic Displays the active toolbox. After specifying a direction from the Direction drop-down list, point to the "handle" (the small black rectangle) at the lower right corner of the toolbox, and drag the toolbox to a new size.

OK button Closes the dialog box and resizes the toolbox.

Cancel button Cancels the command and closes the dialog box.

Help button Displays help for the Button Size dialog box.

Stretch Bitmap dialog box

Use the Stretch Bitmap dialog box to determine how Power Toolbox stretches a bitmap you're pasting into a toolbox.

Stretch Bitmap: list Select a direction to specify whether you want the button bitmaps to stretch horizontally, vertically, both, or not at all.

Toolbox graphic Displays the bitmaps as they will appear after you stretch them to their new size.

OK button Closes the dialog box and resizes the toolbox.

Cancel button Cancels the command and closes the dialog box.

Help button Displays help for the Stretch Bitmap dialog box.

Select All command

The Select All command selects all the toolbox buttons in the toolbox so that you can cut all of the bitmaps and associated commands or copy and paste them into another toolbox. When you paste all the buttons from one toolbox into another, pasted button assignments will replace those in the open toolbox.

To select one button, click the button. To select a block of buttons, drag over adjacent buttons.

Related topics

Power Toolbox window

New... command

The New... command creates a new toolbox within the open toolbox file.

Related topics

New Toolbox dialog box

New Toolbox dialog box

Use the New Toolbox dialog box to create a new toolbox within the open toolbox (.TBL) file. Each toolbox file can contain multiple toolboxes.

Toolbox: box	Type a name for the new toolbox.
Application: box	If desired, type the name of the executable file with which you want to use the toolbox. For example, to associate the new toolbox with the EXCEL.EXE file, type *excel* here.
OK button	Closes the dialog box and displays the new toolbox.
Cancel button	Cancels the command and closes the dialog box.
Help button	Displays help for the New Toolbox dialog box.

Copy... command

The Copy... command copies all buttons in the displayed toolbox to the toolbox that you name.

Related topics

Copy Toolbox dialog box

Copy Toolbox dialog box

Use the Copy Toolbox dialog box to copy all buttons in the displayed toolbox to a toolbox that you name.

Toolbox: box	Type the name you want for the new toolbox, which will contain duplicates of all the buttons in the open toolbox.
Application: box	If desired, type the name of the executable file with which you want to use the new toolbox. For example, to associate the new toolbox with the EXCEL.EXE file, type *excel* here.
OK button	Closes the dialog box and displays the new toolbox.
Cancel button	Cancels the command and closes the dialog box.
Help button	Displays help for the Copy Toolbox dialog box.

Rename... command

The Rename... command renames a toolbox within the open toolbox file.

Related topics

Rename Toolbox dialog box

Rename Toolbox dialog box

Use the Rename Toolbox dialog box to rename a toolbox within the open file.

Toolbox: box	Type a new name for the toolbox.
Application: box	If desired, type the name of the executable file with which you want to use the toolbox. For example, to associate the toolbox with the EXCEL.EXE file, type *excel* here.
OK button	Closes the dialog box and displays the renamed toolbox.
Cancel button	Cancels the command and closes the dialog box.
Help button	Displays help for the Rename Toolbox dialog box.

Delete... command

The Delete... command deletes a toolbox within the open toolbox file.

Related topics

Delete Toolbox dialog box

Delete Toolbox dialog box

Use the Delete Toolbox dialog box to delete a toolbox within the open file.

Delete Toolbox: list Select the toolbox you want to delete from the drop-down list.

OK button Closes the dialog box and deletes the toolbox you specified.

Cancel button Cancels the command and closes the dialog box.

Help button Displays help for the Delete Toolbox dialog box.

Application Associations... command

The Application Associations... command allows you to check or change the associations of applications and toolboxes within the open toolbox file.

Related topics

Applications Associations dialog box

Application Associations dialog box

Use the Application Associations dialog box to check or change the associations of applications and toolboxes within the open toolbox file.

Global Toolbox: drop-down list Select the toolbox you want to use as the global toolbox, which will be available for all Windows applications as long as Power Toolbox is running and the global toolbox is the open toolbox or the Switch with App button is turned on.

Application: box Displays the name of the executable file associated with the toolbox selected in the Toolbox drop-down list. To associate that toolbox with a different application, type the name of the executable file here. For example, to associate the toolbox with the WINWORD.EXE file, type *winword* in this box. Associating toolboxes with applications is optional.

Toolbox: list Shows the name of the toolbox whose association is selected in the **Associations:** list below. To change the toolbox used in that association, select a new toolbox name from this drop-down list, then click the Replace button.

Note: Each application can have only one toolbox associated with it, but toolboxes may be associated with more than one application.

Associations: list Lists applications (by their executable filenames) and the toolboxes with which they are associated.

Insert button Inserts a new association into the **Associations:** list.

Replace button Replaces the association selected in the **Associations:** list with the contents of the Application and Toolbox boxes. For example, you may have selected the association of WRITE and Business Buttons in the **Associations:** list, and then typed *winword* in the **Application:** box to associate the Business Buttons mapping with Word for Windows instead of with Write. Associating toolboxes with applications is optional.

Delete button Deletes the association selected in the **Associations:** list.

OK button Closes the dialog box and accepts the changes you specified.

Cancel button Cancels the command and closes the dialog box.

Help button Displays help for the Application Associations dialog box.

Index command

Displays the index for the Power Toolbox Help system.

Keyboard command

Displays information on using Power Toolbox with keys.

Commands command

Displays help information about Power Toolbox commands.

Using Help command

Displays the Windows Index to Using Help.

About... command

Displays the hDC Power Toolbox version number and copyright information.

Power Toolbox window

Use the Power Toolbox window to create and edit buttons in toolboxes.

To display a floating toolbox, you must minimize Power Toolbox or go to another window. This displays your toolbox, which "floats" on the screen. You can reposition the toolbox by dragging it by its title bar. You can also control whether the toolbox floats on top of all other windows by choosing the Float Toolbox command from the toolbox's Control menu.

Toolbox list Shows the name of the active toolbox. You can choose another toolbox from the drop-down list. To add a new toolbox to the list or to delete a listed toolbox, use commands from the Toolbox menu.

Switch with App button When on, automatically displays the associated toolbox (if any) and the global toolbox side-by-side when you switch to an application. To associate toolboxes and applications, use the Application Associations command on the Toolbox menu. When off, displays only the toolbox shown in the **Toolbox:** list.

Toolbox graphic Select the toolbox button to which you want to add a bitmap or an Enhanced Command. To select more than one button, drag over adjacent buttons.

Capture tool Turns the mouse pointer into a rectangle that will copy any part of the screen. To copy a screen bitmap and paste it into a toolbox, make sure the bitmap is visible on the screen, click the button in the toolbox graphic where you want to place the picture, click the capture tool to turn the mouse pointer into a capture rectangle, then position the mouse pointer over the bitmap and click.

Cut button Removes the contents of the selected toolbox buttons.

Copy button Copies the contents of the selected toolbox buttons.

Paste button Pastes the contents of the Clipboard into the selected toolbox buttons.

Clear button Removes the contents of the selected toolbox buttons.

Command: box Shows the Enhanced Command you are assigning to the selected toolbox button. You can create a command by using the Command Builder, typing directly into the **Command:** box, or by recording a macro with the Record button. To edit text in the **Command:** box, click in the text, insert new text, or delete text by selecting it and pressing the DELETE key. To add a comment to an Enhanced Command, type a space, a pipe symbol (|), another space, and then your comment. If multiple commands appear in the **Command:** box, add comments between the command and the semicolon separator.

Check button Accepts the command displayed in the **Command:** box, assigning the command to the key or key combination selected in the keyboard graphic above.

Cancel button	Removes the command displayed in the **Command:** box and restores the old command (if any).
Command Builder button	Displays the Command Builder window.
Record button	Starts the recorder.

Recorder button

Starts the recorder.

Cancel button

Removes the command displayed in the **Command:** box and restores the old command (if any).

Check button

Accepts the command displayed in the **Command:** box, assigning the command to the key or key combination selected in the keyboard graphic above.

Capture tool

Turns the mouse pointer into a rectangle that will copy any part of the screen.

Cut button

The Cut button removes the contents of the selected toolbox buttons.

Copy button

The Copy button copies the contents of the selected toolbox buttons.

Paste button

The Paste button pastes the contents of the Clipboard into the selected toolbox buttons.

Clear button

The Clear button removes the contents of the selected toolbox buttons.

Using Power Toolbox with a keyboard

Use the following keys within dialog boxes. A plus sign between key names means to hold down the first key while you press the second key.

Press	To
TAB	Move from option to option (left to right and top to bottom).
SHIFT+TAB	Move from option to option in reverse order.
ALT+LETTER	Move to the option or group whose underlined letter matches the one you type.
ARROW KEYS	Move the selection cursor from option to option within a group of options. - or - Move the selection cursor left, right, up, or down within a list box or text box.
HOME	Move to the first item or character in a list or text box.
END	Move to the last item or character in a list or text box.
PGUP or PGDN	Scroll up or down in a list box, one window at a time.
ALT+DOWN ARROW	Open a drop-down list box.
SPACEBAR	Select or cancel a selection in a list box. Also, to select or clear an option.
ENTER	Carry out the currently selected options and list items in a dialog box.
ESC	Close the dialog box without completing the command.

hDC Command Builder

The Command Builder provides an easy way to construct Enhanced Commands without knowing any special command syntax. Although you may not use the Command Builder every time you create a command, it simplifies the process of building complex commands. The Command Builder is usually accessed by clicking the Command Builder button.

The Command Builder contains seven buttons, described below.

Applications & Documents Lets you select the executable and document files you want to launch. See Applications & Documents

MicroApps Lets you select the MicroApps from hDC and other software vendors you want to launch. See MicroApps

Scripts Lets you select the pre-defined scripts you want to launch. See Scripts

Replacement Variables Lets you create dialog boxes that prompt you for a filename or parameter when you launch an application. It also lets you access any of the environment variables currently set on your system. See Replacement Variables

Enhanced Command Parameters Lets you control the way an application is launched, including its size, position on the screen, and much more. See Enhanced Command Parameters

Enhanced Command History Displays the last 20 Enhanced Commands you've run using the Launch... command. See Enhanced Command History

Macro Recorder Lets you record macros containing keystrokes and mouse actions. See Macro Recorder

The Command Builder also contains two dialog boxes.

MicroApp Path dialog box
Script Library Path dialog box

Applications & Documents

This portion of the Command Builder lets you select applications and/or documents to launch. When you click the Applications & Documents button (or press ALT+A), the Command Builder displays the options described below.

Show box	Specifies the files displayed in the **Files** box. To change the display, type the new filename (including wildcard characters, if you want) and press ENTER or click the Check button. You can also use this box to change the current path: simply type the new drive or directory name, and press ENTER or click the Check button.
Check button	Updates the **Files** and/or **Directories** boxes to reflect the contents of the **Show:** box.
Directory	Displays the current drive and directory.
Files box	Displays the files contained in the current directory. Use the mouse to select as many files as you want. To select files that aren't adjacent (for example, the third and fifth files in the list), hold down the CONTROL key and click the mouse.
Directories box	Changes drives or directories. To return one level toward the root directory, click "[..]".
Use Full Path option	Builds the command using the full pathname. Turn this option on if you've selected an executable or document file that is not on your DOS search path.

Related topics

Command Builder
Syntax for launching applications and documents

MicroApps

This portion of the Command Builder lets you select the MicroApps you want to launch. It also gives you a brief description of each MicroApp, and lets you set the MicroApp path. When you click the MicroApps button (or press ALT+M), the Command Builder displays the options described below.

MicroApps box	Lists the MicroApps currently on the **MicroApp Path**. Use the mouse to select the MicroApps you want to launch. To select MicroApps that aren't adjacent (for example, the first and third MicroApps in the list), hold down the CONTROL key and click the mouse.
Describe button	Displays the version number and a brief description for the selected MicroApp.
MicroApp Path button	Displays the MicroApp Path dialog box.

Related topics

Command Builder
Syntax for launching applications and documents

Scripts

This portion of the Command Builder lets you select and launch pre-defined scripts. When you click the Scripts button (or press ALT+S), the Command Builder displays the options described below.

Scripts box Lists the scripts currently on the **Script Library Path**. Use the mouse to select the scripts you want to launch. To select scripts that aren't adjacent (for example, the third and fifth in the list), hold down the CONTROL key and click the mouse.

Describe button Displays a brief description of the selected script.

Library Path button Displays the Script Library Path dialog box.

Reference By buttons Determines how the scripts appear in the **Scripts** box. **Script** lists the scripts by name; **Library and Script** lists them by library and script name. The **Library and Script** option is useful if you've got more than one script library and want to determine the scripts contained in any given library.

Sort By Library option Sorts the scripts according to the order they appear on the **Script Library Path** in the Script Library Path dialog box. This is useful if you've got multiple script libraries and want to see the order in which they will be accessed.

Related topics

Command Builder
Script synta

Replacement Variables

This portion of the Command Builder lets you create dialog boxes that prompt you for a filename or other command parameter when you launch an application. It also lets you access any of the environment variables currently set on your system. When you click the Replacement Variables button (or press ALT+V), the Command Builder displays the options described below.

File Prompt edit controls

Create a dialog box that prompts you for a filename when you launch an application. **Wildcard** controls the type of files displayed in the dialog box. To specify multiple wildcards, separate them with a semicolon. **Caption** sets the text that appears in the dialog box's title bar. The Check button adds the contents of these boxes to the **Command:** box.

IMPORTANT: *As with any command line parameter, you must leave a space after the executable filename; make sure to manually insert a space before the replacement variable.*

For example, you might want to launch Write and bring up a prompt displaying your .WRI and .TXT files. First type *write* in the **Command:** box, followed by a space. Next, type **.wri;*.txt* in the **Wildcard** box and *Select an INI file* in the **Caption** box. Then press the Check button to update the **Command:** box. See also Replacement variable syntax: file().

Parameter Prompt edit controls

Create a dialog box that prompts you for a parameter to pass to the application when it is launched. Use the **Default** box to specify the default parameter. The **Text** box specifies the text string that appears in the dialog box. The Check button adds the contents of these boxes to the **Command:** box.

For example, you might want to launch Word for Windows and pass it a parameter that opens the file you last closed. First, type *winword* in the **Command:** box. Next, type */mFile1* in the **Default** box and *Run this WinWord macro?* in the **Text** box. Then press the Check button to update the **Command:** box. See also Replacement variable syntax: text().

Environment Variables box

Lets you select the environment variables currently defined on your system. To select variables that aren't adjacent, hold down the CONTROL key and click the mouse.

For example, *excel: initdir={env(windir)}* will set Excel's initial directory to the value of the WINDIR environment variable. See also Replacement variable syntax: environment variables.

Other options box

Returns various text strings. **{date}** returns the date in *mm/dd/yy* format. **{time}** returns the time in *hh:mm:ss* format. For example you could create a date and time stamp in a Write document with the command:

write; "Date:{date}<Enter>Time:{time}"

See also Replacement variable syntax: date, time.

{lasttext} returns the text string you last specified from a prompt created with the **Parameter Prompt** controls (described earlier in this topic). In the example cited earlier, for example, **{lasttext}** would return */mFile1*. **{lastfile(dpfe)}** returns the drive, path, file, and extension of the file you last selected from a prompt created with the **File Parameters** controls (described earlier in this topic). For example, you could create a command that inserts a header listing the name of the text file you selected from the file prompt dialog box:

notepad {file(.txt,dpfe,"Select a textfile")};"Filename:{lastfile}"*

See also Replacement variable syntax: {lastfile}, {lasttext}.

Related topics

Command Builder
Syntax for replacement variables

Enhanced Command Parameters

This portion of the Command Builder lets you control the way an application or document is launched, including:

* How the application is displayed: maximized, as an icon, hidden, and more.
* Where the application is launched (absolute or relative position) and the exact size of the application's window.
* What to do if the application is already running: use the existing version, run a new version, and so on.
* What you want to display for the application's window caption and minimized icon.

When you click the Enhanced Command Parameters button (or press ALT+P), the Command Builder displays the options described below.

Show Normal box This drop-down list controls the way the application is displayed after it's launched. See also Enhanced Command Parameters: display logic.

* **Show** launches the application using either the **Icon Maximize** or **Center** parameter, depending on the application. Most applications will use **Icon Maximize**, but non-sizable applications and applications with small windows (like Control Panel and PIF Editor) will use the **Center** parameter.
* **Hidden** launches the application and hides its window.
* **Icon Maximize** runs the application and maximizes its window leaving room for the Power Bar to appear at the top of the screen and icons to appear at the bottom.
* **Power Maximize** runs the application and maximizes its window leaving room for the Power Bar to appear at the top of the screen.
* **Maximized** runs the application and enlarges its window to fill the entire screen.
* **Minimized** launches the application and shrinks its window to an icon.
* **Center** positions the application's window in the center of the screen.

Use Existing box This drop-down list controls the logic used to handle applications that are already running. If you're launching an application that is already running, **Use Existing** will not run a second instance. For example, if you're opening a Write document while Write is already running, using this option loads the document in the existing instance of the application. If the application you're launching is *not* running, the **Use Only Existing** option ignores the command. If the application you're launching is already running, **New Instance** will run a new copy of the program. See also Enhanced Command Parameters: instance logic.

Zoom box This drop-down list controls whether or not the application's window "zooms" when it is launched. Some applications may have problems if **Zoom** is on; if so, use the **No Zoom** option. See also Enhanced Command Parameters: zoom logic.

X and **Y, Width** and **Height boxes**

These determine how an application is sized and positioned when it is launched. You can specify any parameter in absolute terms (expressed in pixels) or relative terms (expressed in a percentage of the screen size). You can specify parameters numerically by typing them, or graphically using the Position button.

X and **Y** Control the vertical and horizontal position of the window. For example, if *X=0* and *Y= 32*, the left (or vertical) border of the application's window will appear on the first pixel, while the top (or horizontal) border will appear on the thirty-third pixel. **Width** and **Height** determine the size of the window, expressed in number of pixels or percentage of screen size. See also Enhanced Command Parameters: window size and position.

Position button

Using the Position button is probably the easiest way to size and position an application's window.

Clicking the Position button brings up the Position and Size Tool window. To size an application's window, point to a border or corner of the Position and Size Tool window and drag it until it is the size you like. To position an application's window, drag the title bar to the location where you want the application to appear. Press ENTER (or choose the OK button if it's visible) to save your changes.

DDE Execute box

Sends an initial DDE Execute string to an application. The DDE topic is "system" and the application's name is the same as the executable filename. For information on the DDE syntax for a specific application, consult the application's documentation. For example, *excel:usee,dde=("[INSERT(1)]")* sends a DDE string to Excel that inserts a column. See also Enhanced Command Parameters: DDE.

Initial Directory box

Determines the directory used to launch an application. You may want to set an application's initial directory to the DOS directory containing the files you want to open. For example, if you store all of your spreadsheet files in one directory, you could set the initial directory for Lotus 1-2-3 so that it defaults to that directory. See also Enhanced Command Parameters: initial directory.

Icon box

Determines the application's icon, allowing you to replace the default icon with one from another application. You can use icons from any file that contains them, including .EXE, .ICO, .ICN, .IL, and .DLL files. To specify an icon, type the filename (and path, if necessary) of the file whose icon you want to use; for example, *clock:icon=bigben.ico.*

If the file contains more than one icon, insert a comma and the number of the icon you want to use. For example, *pbrush:min,icon=(progman.exe, 11);* launches Paintbrush and substitutes an icon from PROGMAN.EXE in place of Paintbrush's icon. See also Enhanced Command Parameters: application icon.

Caption box	Determines the caption for the application's window and the label for the application's minimized icon. Type the caption as you want it to appear. See also Enhanced Command Parameters: caption.

Related topics

Command Builder
Syntax for Enhanced Command Parameters

Enhanced Command History

When you click the Enhanced Command History button (or press ALT+H), the Command Builder displays the last 20 commands you've launched from the **Launch** box in the Power Bar or the Command Builder. Select as many commands as you like.

Related topics

Command Builder

Macro Recorder

This portion of the Command Builder lets you record macros containing keystrokes and mouse actions. Macros are useful for automating repetitive tasks. For example, you might create a macro that launches the Windows Control Panel, opens the Printer setup and sets the page orientation to landscape.

When you click the Macro Recorder button (or pressing ALT+M), the recorder begins recording and displays its control panel. The recorder's control panel "floats" over the active window in order to remain visible. As you record your macro, the Enhanced Command will appear in the control panel. To temporarily stop the recorder, click the Pause button; when you're ready to resume recording, click the Pause button again. To stop the recorder, click the Stop button.

Notes on recording macros

* As with any macro recorder, keyboard macros are generally more reliable than macros containing mouse actions. For that reason, as you record your macro it's best to use keystrokes whenever possible.
* Should you need to position a window (or configure some other portion of Windows) after you've begun recording, press the Pause button to temporarily stop recording; when you're ready to resume recording, just press the Pause button again.
* To stop the playback of a macro, press the ESC key. You'll receive a prompt confirming your decision to stop the macro.
* Keep in mind that you can easily edit your macros. For example, if you've made a mistake while recording a macro, just press the Pause button, move the insertion point into the recorder's control panel and delete the error. When you're ready start recording, press the Pause button again.
* To achieve the best results, you should prepare your screen before you begin recording a macro. If there are windows on the screen that aren't part of the macro, move them out of the way or close them before you begin recording.
* Should you need to slow down the playback of a macro, you can insert one of the [Pause] scripts into a macro.
* The macro recorder does not record certain actions, such as mouse drags. Also, it cannot record mouse actions involving the toolbar in some programs such as Word for Windows.

Related topics

Command Builder
Macro syntax

MicroApp Path dialog box

Use this dialog box to specify where your MicroApps are located; for example, you might type *C:\WINDOWS\MICROAPP*. If your MicroApps are stored in more than one location, separate each path with a semicolon.

Related topics

MicroApps

Script Library Path dialog box

Use this dialog box to specify where your scripts are located, and in what order they should be searched.

Filename(s) box	Specifies the files displayed in the **Files** box. To change the display, type the new filename (including wildcard characters, if you want) and press ENTER. You can also use this box to change the current path: simply type the new drive or directory name, and press ENTER.
Directory	Displays the current drive and directory.
Files box	Displays the files contained in the current directory. Use the mouse to select as many files as you want.
Directories box	Changes drives or directories. To return one level toward the root directory, click "[..]".
Add button	Adds the selected files to the **Script Library Path**.
Remove button	Removes the selected files from the **Script Library Path**.
Script Library Path box	Displays the script libraries currently on the path. Libraries are searched in the order specified in the list. If the same script appears in multiple libraries, the script appearing first on the path is used.
Arrow buttons	Changes the order of the script libraries currently highlighted in the **Script Library Path** box. Script libraries are searched in the order in which they appear in this box.

Related topics

Scripts

Remap Key dialog box

This dialog box lets you select characters from the ANSI character set and paste them onto the Clipboard. You can select a character using either your mouse or the arrow keys on your keyboard. To copy the character to the Clipboard, choose the Select button or press ENTER.

Text Prompt dialog box

This dialog box appears when you've launched a command that includes a parameter prompt. To launch the application with the specified parameter, choose the OK button. To launch the application without a parameter, choose None. To escape without launching the command, choose Cancel.

Related topics

Replacement Variables

Enhanced Command syntax

Although Power Launcher's Command Builder lets you create commands using buttons and other graphical controls, you may find that using the Enhanced Command syntax lets you work more quickly. Additionally, knowing the syntax enables you to do a few things that aren't possible with the Command Builder, such as call a function in a Dynamic Link Library (.DLL) file.

Application and document syntax
DOS command syntax
Enhanced Command Parameter syntax
Dynamic Link Library (.DLL) syntax
Macro syntax
Replacement variable syntax
Script syntax
Adding comments to an Enhanced Command

TIP: *You can use the Copy command from the Edit menu in Windows Help to copy any of the sample commands listed in this section, and then run the command by pasting it into the* ***Launch:*** *box.*

The syntax conventions are listed in the following table.

Convention	Example	Description
Boldface	**useexist**	Text that must be typed exactly as it appears.
Italic	*height*	A general placeholder for information that you provide.
[ ]	[**,** *arg*]	An optional command element. Do not type the brackets themselves (except for the brackets that surround Scripts).
...	[*keystroke*]...	Indicates that the preceding element can be repeated several times. You can repeat any number of keystrokes. Do not type the ellipsis (...) itself.
|	**zoom** | **nozoom**	Separates items in a group of mutually exclusive elements. Choose only one of the items from the group. Do not type the pipe symbol (|) itself (except for the pipe that delineates a Comment).
	`clock:min,usee`	This font is used to illustrate Enhanced Command examples. You can copy commands from online Help, paste it directly into a **Launch:** box, and edit the example to fit your needs.

Application and document syntax

In addition to launching applications and opening documents, you can optionally pass DOS command line parameters and send Enhanced Command Parameters to an application. Enhanced Command Parameters are preceded by a colon and must be separated by spaces or commas:

```
app-spec [app-parms] [: eparm] [ [,] eparm]...
```

Keep in mind that Enhanced Commands also accept replacement variables: parameters and functions that let you specify filenames, parameters, and more.

The app-spec syntax is:

```
[path]filename | replacement-var
```

Example

This example will launch Excel and open the document named Q1SALES.XLS.

```
c:\sales92\q1sales.xls
```

The app-parm syntax is:

```
command-line-parms | replacement-var
```

Example

This example launches Word for Windows and opens the last document you closed.

```
winword /mFile1
```

Related topics

Enhanced Command syntax

DOS command syntax

You can run several "internal" DOS commands from an Enhanced Command, without running a DOS session. The DOS commands are listed below. For complete information on DOS commands, see your *DOS User's Guide.*

Command...	**What it does:**
copy	Copies a file.
move, mv, rename, ren	Moves or renames a file.
delete, del, erase	Erases a file.
mkdir, md	Creates a directory.
chdir, cd	Changes the current directory.
rmdir, rd	Deletes a directory.
ddel	Deletes a directory and any files or subdirectories it contains.
dir	Uses Windows File Manager to display the current directory contents.

Notes

* Be careful when deleting files or directories: you will not receive a confirmation prompt!
* To display the current directory contents using a DOS box instead of the Windows File Manager, add the line *DOSBOXDir=1* to the [File Manager] section of the HDC.INI file.

Related topics

Enhanced Command syntax

Enhanced Command Parameter syntax

In addition to launching an application and specifying DOS command line parameters, you can also send it Enhanced Command Parameters. These parameters are preceded by a colon and may be separated by spaces or commas:

```
app-spec [app-parms] [: eparm] [ [,] eparm]...
```

Enhanced Command Parameters fall into several logical groups, many of which contain mutually exclusive options; for example, a window cannot be both **min**imized and **max**imized. If conflicting options are specified, the parameter appearing last will override any that precede it. You can substiture unambiguous abbreviations (such as **h** for **hidden** and **usee** for **useexist**) for parameters.

You can set default Enhanced Parameters--globally or for specific applications--in the HDC.INI file. The global defaults are **useexist**, **show**, **usee, zoom**, and **env=full**.

Enhanced Command Parameters

caption
close
dde
env
forcesize
hidden
icon
imax
initdir
keys
max
min
newinst
nozoom
pos
show
useexist
useonlyexist
wait
zoom

Related topics

Enhanced Command syntax

Enhanced Command Parameters: instance logic

useexist | useonlyexist | newinst

Determine what to do if you try to launch an application that is already running. The **useexist** parameter tries to send the command to an application that is already open; if the application is not open, the application is launched. **useonlyexist** forces the Enhanced Command to use an existing instance of the application; if none is open, the command does not run. **newinst** launches a new instance of the application. The shortest unique abbreviations are **usee**, **useo**, and **ne**, respectively.

Example

This example will activate the existing instance of Windows Write and change its caption to "Testing." If Write is not open, it will launch the application.

```
write: usee,caption="Testing"
```

Related topics

Enhanced Command Parameter syntax

Enhanced Command Parameters: display logic

show | hidden | max | imax | min | center

Determine how the application window is displayed. Each parameter is explained below:

Parameter...	Abbreviation	Displays the window:
show	**s**	Displays the window using either **imax** or **center**, depending on the application. Most applications will use **imax**, but non-sizable applications and applications with small windows (like Control Panel) will use **center**.
hidden	**h**	Hidden.
max	**ma**	Maximized.
imax	**im**	Maximized with room for the Power Bar at the top of the screen and icons at the bottom of the screen.
min	**mi**	Minimized.
center	**ce**	Centered.

Examples

This example will launch Word for Windows and hide its window.

```
winword:h
```

This example will launch CorelDRAW and maximize its window, leaving room for icons at the bottom of the screen.

```
coreldrw: im
```

Related topics

Enhanced Command Parameter syntax

Enhanced Command Parameters: zoom logic

zoom | nozoom

Determine whether or not the application window "zooms" when it is launched. The shortest unique abbreviations are **z** and **noz**.

Example

This example will launch the Windows Control Panel and zoom its window.

```
control: zoom
```

Related topics

Enhanced Command Parameter syntax

Enhanced Command Parameters: closing an application

close

Closes an application. The **noclose** parameter overrides **close**. The shortest unique abbreviations are **cl** and **noc**.

Example

This example will close an existing instance of Write.

```
write: useonlyexist, close;
```

Related topics

Enhanced Command Parameter syntax

Enhanced Command Parameters: application caption

caption= "*caption***" | default**

Sets the caption for the application's window and icon. The caption may include replacement variables. Choosing **default** overrides any caption setting in HDC.INI and allows the application to set its own caption. The shortest unique abbreviation is **ca**.

Example

This example launches Notepad and inserts the date into Notepad's caption.

```
notepad:caption="Opened on {date}"
```

Related topics

Enhanced Command Parameter syntax

Enhanced Command Parameters: initial directory

initdir=*initial-directory*

Sets the initial directory for the application and for any replacement variables. The shortest unique abbreviation for **initdir=** is **in=**.

Example

The following example launches Excel and sets the initial directory to C:\BUDGET.

```
excel:initdir=C:\BUDGET
```

Related topics

Enhanced Command Parameter syntax

Enhanced Command Parameters: window size and position

pos= ([*x*[**%**]]**,** [*y*[**%**]] [**,** [*width*[**%**]]**,** [*height*[**%**]]] **)** | **default**

Positions and sizes the application window. Each parameter must be an integer and can be expressed in absolute terms or as a relative percentage of the virtual screen size. Specifying parameters that are outside the boundaries of the current screen will position the application's window in another portion of the virtual desktop. Choosing **default** allows the application to position and size itself. The shortest unique abbreviation for **pos=** is **p=**.

Examples

The following example launches Paintbrush and positions it on the virtual desktop to the right of the upper left-most screen. This command positions the window in absolute terms based on the display resolution.

```
pbrush:pos=(640, 0)
```

If you frequently change your video display driver resolution, you should position your application windows using the method shown in the next example. Unlike the previous example, this command positions the window as an absolute percentage of the virtual screen size.

```
pbrush:pos=(100%, 0)
```

Related topics

Enhanced Command Parameter syntax

Enhanced Command Parameters: keystrokes

keys= "[*keystroke*]..."

Sends any number of keystrokes (such as a string of characters or a macro command as described later under key-stream) to the application. Unlike the key-stream Enhanced Command used in the macro recorder--which sends keystrokes to the application currently active--this parameter sends keystrokes to a specific application. In addition, it is the only way to send keystrokes to a hidden application. The shortest unique abbreviation for **keys=** is **k=**.

Example

The following example send the text string enclosed in quotes to Word for Windows.

```
winword:keys="From the desk of Jane Doe"
```

Related topics

Enhanced Command Parameter syntax

Enhanced Command Parameters: application icon

icon= (*ICON-filespec*[**.ico**] | *IL-filespec* [**,** *icon-number*] **)**

Allows you to specify the icon you want to display for a minimized application. "ICON-filespec" refers to an icon file (.ICO) containing a single icon. "IL-filespec" may be any file that contains an icon, such as an hDC Icon Designer library (.IL), an application (.EXE), or a Dynamic Link Library (.DLL). If "icon-number" is not specified, the first icon in the file is used. The shortest unique abbreviation for **icon=** is **ic=**.

Example

The example below launches Notepad, which when minimized will display the 18th icon from the file named PROGMAN.EXE.

```
notepad: icon= (progman.exe, 18 )
```

This example launches Solitaire, which when minimized will display the icon for Windows Write.

```
sol.exe: icon=(c:\windows\write.exe)
```

Related topics

Enhanced Command Parameter syntax

Enhanced Command Parameters: environment

env= full | **(***envvar* [**=** [*value*]] [**,***envvar* [**=** [*value*]]]... **)**

Sets the application's environment. The default environment is **full**, which provides the application with the complete DOS environment.

You can use the **env=** parameter in an Enhanced Command to minimize the launch environment, thus reducing the memory that is used by an application. For example, you can specify an environment that contains only the certain variables from the DOS environment by leaving its value field empty; for example, *env=(COMSPEC, PATH=)* specifies that COMSPEC is to be included, and that PATH is to be removed. The specified environment cannot be larger than the original DOS environment.

The shortest unique abbreviation for **env=** is **e=**.

Related topics

Enhanced Command Parameter syntax

Enhanced Command Parameters: forcesize

forcesize

Some applications don't allow themselves to be sized or positioned if, during their last session, they were closed while the window was maximized; examples include Microsoft Excel and Windows File Manager. Other applications (such as Lotus 1-2-3) do not allow themselves to be sized or positioned at all. The **forcesize** parameter allows you to size and position such applications. The **noforcesize** parameter will override **forcesize** (if **forcesize** were the HDC.INI default, for example). The shortest unique abbreviations are **f** and **nof**.

Example

The following example runs Lotus 1-2-3 sized and positioned:

```
123w.exe:imax,f
```

Related topics

Enhanced Command Parameter syntax

Enhanced Command Parameters: wait

wait

In an Enhanced Command that launches multiple applications, this parameter waits until one application is finished before processing the next part of the Enhanced Command. This is particularly useful when running DOS commands. The parameter **nowait** runs the command normally. The shortest unique abbreviations are **w** and **now**.

Example

The following example runs A.BAT, waits until it is finished, then runs B.BAT:

```
A.BAT:wait;B.BAT
```

Related topics

Enhanced Command Parameter syntax

Enhanced Command Parameters: DDE (Dynamic Data Exchange)

dde= ("*execute-string*" [**,** "*topic*" [**,** "*application*"]] **)**

Sends *execute-string* as a Dynamic Data Exchange execution to an application. The DDE "application" name defaults to application's filename, and the topic defaults to "system." The shortest unique abbreviation for **dde=** is **d=**.

Example

This example sends a DDE string to Excel that inserts a column. You could perform the same action using hDC's macro recorder, but using DDE is slightly faster and less obtrusive, since you don't see the menu drop down, the dialog box open, etc.

```
excel:s,usee,z,dde=("[INSERT(1)]")
```

This example takes a selected, tab-delimited table in a word processing application such as Write or WinWord, charts it with Excel (which remains hidden), and pastes the chart into the original document.

```
"<Menu=Edit-Copy>"; excel: h, dde=("[PASTE()][NEW(2,2)][COPY()]
[CLOSE(0)][CLOSE(0)]"), close; "<Down><Menu=Edit-Paste>"
```

Related topics

Enhanced Command Parameter syntax

Dynamic Link Library (.DLL) syntax

An Enhanced Command may call a function from a Dynamic Link Library (.DLL) file. You can use the shortcut **win.dll** to call any Windows function contained in USER.DLL, GDI.DLL, or KERNEL.DLL.

*dll-spec***.dll** *procname***(** [*dll-arg*[**,** *dll-arg*]...] **)**

dll-arg

You can pass arguments to a DLL function using a decimal or hexadecimal number, or using a string. The specific number and type of arguments depend upon the procedure.

[**-**]*decimal-num*[**L**] | **0x***hex-num*[**L**] | **"***string***"**

The **L** must follow decimal-num or hex-num if and only if the function requires a long integer argument.

Examples

The following example will sound a system message beep.

```
win.dll MessageBeep(0)
```

The next example calls a function in hDC's Dynamic Link Library (HDCLIB.DLL) that displays a message box.

```
hdclib.dll hDCMessageBox(0,"Your message goes here","Caption",0,0L,0L)
```

The last example will restart Windows. (Calling this function ends the current Windows session immediately without displaying a confirmation prompt.)

```
win.dll ExitWindows(0x42L, 0)
```

Related topics

Enhanced Command syntax

Adding comments to Enhanced Commands

Each Enhanced Command may include a comment. Comments must be preceded by a pipe symbol (|). Any text between the pipe and the semicolon that terminates the command will be ignored. For example:

```
notepad: min, newinst | starts minimized notepad;
```

Related topics

Enhanced Command syntax

Macro syntax

Macros use a *key-stream,* which is a sequence of *keystrokes* surround by quotation marks:

"*keystroke*[*keystroke*]...**"**

Each *keystroke* may be a printable character, a command combination, which is enclosed in **<**angle brackets**>,** or a replacement variable. Listed below are the conventions used to translate keyboard and mouse actions into Enhanced Command syntax.

Keystroke convention	Convention	Example
Character		
character		*notepad:"hello"*
Non-printing keypress		
<*key-label***>**	**<Enter>**, **<F7>,<Space>**	*write:"Name<Enter>Address<Enter>"*
Modifier + key		
<*modifier* **-** *key***>**	**<Alt-a>**, **<Ctrl-Shift-End>**	*control;"<Alt-S><Enter>"*
Menu selection		
<Menu=*choice*[**-***choice*]...**>**	**<Menu=File-Open>**	*pbrush;"<Menu=File-Open...>"*
<Sysmenu=*choice*[**-***choice*]...**>**	**<Sysmenu=Exit>**	*excel;"<Sysmenu=Restore>"*
Button press		
<Button=*button-name***>**	**<Button=OK>**	*charmap;"<Button=Select>"*
Move window		
<Move=*title***-***xpos***,***ypos***>**	**<Move=Notepad-200,200>**	*clock;"<Move=Clock-200,200>"*
Size window		
<Size=*title***-***width***,***height***>**	**<Size=Notepad-400,250>**	*clock;"<Size=Clock-400,250>"*
Mouse click		
<Mouse[**-***pos*]=[*modifier***-**]*click***>**	**<Mouse=Ctrl-Shift-LeftDbl>**	*pm4;"<Mouse=Right>"*

The characters ( ) [ ] { } < > | % ^ - = ; : have particular meanings within an Enhanced Command. If you want to include one of these special characters within a keystream, precede it with a caret symbol (^). For example, *write; "hDC ^"The Windows Specialists^"."*

Related topics

Enhanced Command syntax

Replacement variable syntax

Replacement variables let you create dialog boxes that prompt you for a filename or parameter when you launch an application or document. They also lets you access any of the environment variables currently set on your system. Replacement variables take the form:

{ *variable* | *function***(** [*arg*[**,** *arg*]...] **) }**

You can use replacement variables when you launch an application, call a .DLL function, or send a **keys=**, **caption=**, **icon=**, **initdir=**, **dde=**, or **env=** parameter. The replacement variable functions are listed below.

Replacement variables

{file()}
{text()}
{lastfile}, {lasttext}
{date}, {time}
{env()}

Related topics

Enhanced Command syntax

Replacement variable syntax: {file()}

This function displays a dialog that asks you to select a file. The syntax is:

{file(*wildspec [;wildspec]...* [**,** [**d**] [**p**] [**f**] [**e**] [**,** "*prompt*"]] **)}**

Wildspec determines which files (such as **.txt*) are displayed in the **Files** box. Note that you can specify multiple wildcards. The caption for the dialog box's window is set by the *prompt*, which defaults to **Choose document for** <*appname*>. The four parameters, **d**, **p**, **f**, and **e**, determine the portion of the pathname returned by **{file()}**. A fully-qualified filename (**dpfe**) is the default.

Parameter:	What it does...
d	Requests the drive, including a trailing colon.
p	Requests the path, including a trailing slash if used together with the **f** parameter.
f	Requests the filename.
e	Requests the extension, including a leading period if used in conjunction with **f** parameter.

Example

The following command will display a prompt containing the .WRI and .TXT files in the current directory.

```
Write { file( *.wri;*.txt, dpfe, "Choose a document" ) }
```

Related topics

Replacement variable syntax

Replacement variable syntax: {text()}

This function asks you to supply a string, which is substituted into the Enhanced Command. The syntax is:

{text("*default-text*" [**,** "*prompt*"] **)}**

It is important to note that Power Launcher does not check or interpret the syntax of the string you've specified. The default prompt is **Parameters for** <*appname*>.

Example

This command will bring up a text prompt specifying */mFile1* in a text box as the default parameter for WinWord (the parameter launches a WinWord macro that opens the document last opened).

```
winword.exe {text("/mFile1","Run this macro?")}
```

Related topics

Replacement variable syntax

Replacement variable syntax: {lastfile}, {lasttext}

The **{lastfile}** function returns the drive, path, file and/or extension of the file you've most recently selected using **{file()}**. **The syntax is:**

{lastfile} [**(** [**d**] [**p**] [**f**] [**e**] **)**]

{lasttext} returns the text most recently specified via the **{text()} function. If you haven't called the {text()}** function, no value is returned.

Example

This command inserts the name of the text file you selected from the file prompt dialog box.

```
notepad {file(*.txt,dpfe)};"Filename: {lastfile}"
```

Related topics

Replacement variable syntax

Replacement variable syntax: {date}, {time}

{date} returns the current date in *mm/dd/yy* format. **{time}** returns the current time in *hh:mm:ss* 24 hour format.

Example

The following command creates a date and time stamp in a Notepad document.

```
notepad; "DATE:{date}<Enter> TIME:{time}"
```

The next command launches Calendar and set its caption to the current date.

```
calendar:cap="{date}";
```

Related topics

Replacement variable syntax

Replacement variable syntax: environment variables

You can insert environment variables into an Enhanced Command. Percent signs surround the variable you want to replace:

{env *(ENVVAR)* **}**

Example

This example sets Write's initial directory to the value of WINDIR .

```
write: initdir={env(WINDIR)}
```

The next example changes the directory to the value of WINDIR (that is, changes to the Windows directory).

```
cd {env(WINDIR)};
```

Related topics

Replacement variable syntax

Script syntax

Enhanced Commands can also execute the pre-defined scripts included with Power Launcher. If the library (.HSL) file containing the script is not on the Script Library Path, a command must explicitly reference the pathname. The entire script reference must be surrounded by **[** square brackets **]**.

[[*scriptlib***!**]*scriptname* **]**

Example

The example will launch a script that minimizes all applications that are currently open.

```
[Minimize All]
```

The next example will launch a script that cascades all currently-open applications.

```
[sizing.hsl!Cascade All]
```

Related topics

Enhanced Command syntax

Enhanced Command

A superset of the Windows command line that lets you launch applications, documents, MicroApps, scripts or macros. Enhanced Commands may also contain enhanced parameters or replacement variables.

Command Builder button

Brings up the Command Builder for you to create an Enhanced Command using icons, buttons, and other graphical controls.

Applications & Documents button (ALT+A)

Lets you select applications and/or documents to launch.

MicroApps button (ALT+M)

Lets you select the MicroApps you want to launch. It also gives you a brief description of each MicroApp, and lets you set the MicroApp path.

Scripts button (ALT+S)

Lets you select and launch pre-defined scripts.

Replacement Variables button (ALT+V)

Lets you create dialog boxes that prompt you for a filename or parameter when you launch an application or document. It also lets you access any of the environment variables currently set on your system.

Enhanced Command Parameters button (ALT+P)

Lets you control the way an application or document is launched, including its appearance (hidden, minimized, and so forth), size and position, window caption, icon, initial directory, and more.

Enhanced Command History button (ALT+H)

Displays the last 20 commands you've launched from the Launch Command Line dialog box.

Macro Recorder button (ALT+R)

Starts the macro recorder and brings up the recorder's control panel.

Position button

Appears in the Command Builder when you click the Enhanced Command Parameters button. The Position button brings up a window for you to graphically set the size and position an application uses when launched.

Pause button

Temporarily stops the macro recorder.

Stop button

Stops the macro recorder and returns you to the Command Builder.

Check button

Updates the contents of an edit control such as the **Files** box or **Command:** box.

foo

- Application and document syntax
- Enhanced Command Parameter syntax
- Dynamic Link Library (.DLL) syntax
- Replacement variable syntax
- Script syntax
- Adding comments to an Enhanced Command