



## SmilerShell Help

Choose a topic on which you'd like more information:

[Keyboard Shortcuts](#)

[Quick Start / Hints And Tricks](#)

[How To Order SmilerShell](#)

[Introduction](#)

[Why Is This A Shell?](#)

[Installing SmilerShell](#)

[Uninstalling SmilerShell](#)

[What Happens When You Start SmilerShell](#)

[Right-Click The Button: The Apps Menu](#)

[Menu Items](#)

[Submitting Commands](#)

[Command History Button: Lists All Commands](#)

[Using Arrows To Retrieve Previous Commands](#)

[Editing Commands](#)

[The FindFile Dialog](#)

[Size Of Window](#)

[Getting Rid Of Inactive Windows](#)

[DOS Commands: Fullscreen Or Windowed](#)

[About Internal DOS Commands](#)

[Aliases](#)

[The Settings Dialog: Set Preferences](#)

[Command Stack Files](#)

[Choosing Fonts And Colors](#)

[The Initialization File](#)

[Using SmilerShell On A Network](#)

[The ONECMD Option: Set Up Custom Desktop Icons To Do Any Task](#)

[Built-In SmilerShell Commands](#)

[ALARM: Alarm Clock With Reminder Message](#)

[ALIAS and UNALIAS: Create, Change, Remove Aliases](#)

[DC: Directory Change The Fast Way](#)

[FIND: Search For Files On Disk](#)

[HISTORY: Display Command History](#)

[PATH: Change The Search Path](#)

[SHOW: An Alternative To DIR](#)

[Using 4DOS/NDOS Internal Commands](#)

[Notices](#)

## Introduction

Windows is great, but sometimes it can be awkward. Getting to the icon to launch an application isn't always easy. Finding files is difficult at best. There's no good way to keep an eye on important systemwide resources. And even when you have all the pieces, sometimes it'd be faster to just type a command -- but how?

That's what SmilerShell is for.

SmilerShell is a compact yet powerful Windows control center that takes *no* space on your desktop. Until you need it, SmilerShell is just a tiny button that hops into the titlebar of whichever app you're working in. Right-click the button to see your configurable "Apps" list and launch a new program, or switch to a currently-running task. Or left-click the button to reveal the ultimate Windows command line, which supports pipes, redirection, and internal DOS commands (and of course runs Windows programs too).

SmilerShell has the best command line you've ever seen, as if the plain-vanilla DOS prompt was enhanced by lots of handy utilities. It runs anything (DOS programs, Windows programs, or DOS internal commands), supports pipes and redirection, and you can run multiple commands on one line. It has a built-in command line editor with history and search, aliases (type-in or on the function keys), a built-in fast-directory-change utility that works across multiple drives, and many helpful Windows functions like a calendar/clock in the title bar and a real-time system resources report in the menu bar. After it pops up it's still a very compact window, but to make it even smaller you can toggle away the menu or title bar.

SmilerShell's built-in 'find' function lets you search for a file by name, or contents, or both. It lists all matching files and lets you either go to a file's directory, or fetch its name into the command line, ready to run, or perhaps to use as another program's parameter.

SmilerShell's aliases are short commands that are replaced with longer commands. Aliases can be like regular commands, just type them in. Or they can be on function keys, hit the F-key and it runs, no need to press Enter.

SmilerShell's fast directory-change utility is called DC. Just type DC and the first few characters of the directory you want to be in, and SmilerShell takes you there. If your command is ambiguous, a window pops up, letting you choose which directory you want. This works across as many multiple drives as you tell it to be aware of.

These are just a few of the features of SmilerShell. The **Quick Start** section has a description of each major feature. And of course all features are documented in the individual sections of this manual.

So, welcome to SmilerShell! You're going to like it here.

**Related Topics:**

[Quick Start / Hints And Tricks](#)

[Why Is This A Shell?](#)

[The Initialization File](#)

[Menu Items](#)

[Submitting Commands](#)

[Using Arrows To Retrieve Previous Commands](#)

[Aliases](#)

[DC: Directory Change The Fast Way](#)

[The Apps Menu](#)

[Command Stack Files](#)

## How To Order SmilerShell

Thank you for trying SmilerShell. You are welcome to test the fully-functional evaluation version for 30 days. That is, you can run the program on 30 different dates. These dates do not have to be consecutive calendar days. If you don't run SmilerShell on a particular date, it doesn't count against your 30 days.

The evaluation version of SmilerShell contains all the features found in the actual product. You have plenty of time to try it under actual working conditions on your own system, to see if it meets your needs. After the trial period, you must either purchase SmilerShell or remove it from your system.

To order, send \$29.95, plus \$3.50 for shipping (cash, check, money order, MasterCard, or Visa) to:

**Bardon Data Systems**  
**1023 Key Route Blvd.**  
**Albany, CA 94706**

**Outside North America** please add an additional \$6 overseas shipping surcharge.

With a MasterCard or Visa you can also order SmilerShell by phone, either directly from Bardon at **(510) 526-8470** (weekdays 9 to 5 California time) or through our toll-free telephone order-taking service **(800) 242-4775** (weekdays 7 to 6 Central time), or 24 hours a day by fax at **(713) 524-6398** or on CompuServe at **72340,375**. Include card number, expiration date, and name as it appears on the card.

SmilerShell is also available through distributors worldwide. Details, prices, and addresses are in the file DEALERS.TXT that came with this package.

If ordering through Bardon, you can print and mail **orderfrm.wri**, the order form that came with this package. It's in Windows Write format. If you order with a MasterCard or Visa by phone at **(510) 526-8470**, you'll be given your registration number immediately so you can get rid of those reminder screens right away. Or simply mail in your card number and expiration date.

When you order, you'll get a copy of the most recent version of SmilerShell, a registration number that will turn off the reminder screens, a printed manual, support, and update notices.

You will also get a **FREE** copy of **Reverend Lowell's Treasury of Humor, Volume 1: "He Who Laughs, Lasts!"** Reverend Lowell's Treasury contains thousands of humorous anecdotes collected by Reverend Lowell during his long career as an ordained minister. After thirty years of public speaking from the pulpit, the podium, and on television, he has amassed over 200,000 items, and Reverend Lowell's Treasury represents the "cream of the crop." These are Reverend Lowell's favorite

funny stories, revealing quotes, interesting anecdotes, and zingy one-liners. And there's nothing that would be unsuitable in any public setting. There are plenty of items to choose from. Each volume of the Treasury covers a different topic, and contains at least as many items as you'd find in a full-size book. Many are illustrated in color or black-and-white. You can search for any item by keyword or first line. Or simply browse through every item page by page.

In addition, you will get these extra **FREE** bonuses: Two more handy Bardon programs (WHATSNEW lists files not yet backed up, or directories containing such files; PR and PRFILTER format output for printing, and add an informative header), discount certificate for JCSM CD-ROMs at half price, discount on PsL software-by-mail (up to 2/3 off!), free CompuServe startup kit, and whatever other goodies I can fit on the disk.

## What Happens When You Start SmilerShell

When you start SmilerShell, it first reads its initialization file, which stores your SmilerShell preferences as you indicated in the [Settings dialog](#) and through menu switches. By default this is the file "smishell.ini" in the same directory as the SmilerShell program. However, you can specify a name and directory for this on the SmilerShell icon's 'Properties' command line. This is especially useful on a network, where you might have one copy of the program on a server, and many users who want to maintain their own individual setup preferences. See [Using SmilerShell On A Network](#).

Values are set, based on the entries in the initialization file. The list of commands you issued last time is restored (handy for searching instead of typing) as well as any command-stack file you have set up.

If you start SmilerShell without an initialization file, you'll be asked if you'd like SmilerShell to create one and fill it with reasonable initial values. You can change these settings at any time with the [Settings dialog](#) on the [Options](#) menu.

Next, SmilerShell looks for [DC](#) information. It will rescan your directory structure if you've set this in the [Settings](#) dialog. If not, SmilerShell looks for the file it creates when you [Scan Directories](#). By default it is named "smishell.dir" and is in the same directory as the initialization file. You can specify a different name and directory for it in the [Settings](#) dialog.

Then SmilerShell gets the current PATH from the operating system. It uses this to run your commands. If you like, you can change SmilerShell's copy of the PATH. See [PATH: Change The Search Path](#) for details.

If you have set **Save State** in the [Settings](#) dialog, SmilerShell sets itself up the way you left it last time, and changes to the directory you were in when you last used it.

Finally, if you have set **Hide SmilerShell** in the [Settings](#) dialog, SmilerShell hides the command line window, and if you have set **Show Button**, it puts a little button into the title bar of your currently-active application. This button will follow you around to whatever application you activate, so SmilerShell is always just a mouse click away. (Note: occasionally a Windows window is set up to be "always on top." These "topmost" windows will set themselves on top of SmilerShell's button, hiding it. The button will reappear when you switch to a non-topmost window.)

### **Related Topics:**

[The Settings Dialog](#)

[DC: Directory Change The Fast Way](#)

## Menu Items

SmilerShell has five top-level menu items: **File**, **Edit**, **Options**, **Apps**, and **Help**. If the titlebar button is not disabled, there is a **Hide!** menu item, which hides the command line window. Using the **Options** menu, you can also toggle another item onto the menu bar: a real-time report of available Windows memory and resources.

The evaluation version of SmilerShell has a **How To Order** menu item which provides ordering and purchase information.

### **Related Topics:**

[The File Menu](#)

[The Edit Menu](#)

[The Options Menu](#)

[The Apps Menu](#)

[The 'How To Order' Menu Item](#)

[The Hide! Menu Item](#)

[The Help Menu](#)

[The System Resources Menu Item](#)

[Keyboard Shortcuts](#)

## The File Menu

The **File** menu starts with the traditional items **New**, **Open**, **Save**, and **Save As**. These items let you manipulate your current command stack (list of stored commands). Next on the **File** menu are the items **DOS Box**, **Run**, **Aliases**, **Find File**, **Command History**, **Scan Directories**, and of course **Exit**. Here's what these commands do.

**New** clears the command stack. That is, it makes your list of previously issued commands go away. This lets you restart your command history.

**Open** lets you choose a command-stack file and read it in. It clears the current command history stack, and reads in a new command stack from the file. SmilerShell assumes that it's an ASCII text command-stack file, with one command on each line.

**Save** saves your current command stack to a file using the most recently used command-stack file name. The commands are saved in a plain text ASCII file, one command per line.

You set the command-stack file name (to be used by the **Save** item) by specifying one in the **Open** or **Save As** dialogs during the current session, or by setting the **Startup Commands File** item in the **Settings** dialog. If you haven't explicitly set a name yet, the name "smishell.stk" is used. By default it's assumed to be in the same directory as the ini file. If the file doesn't exist, of course no commands are read from it! But a missing file won't cause an error, either.

**Save As** asks for a filename, then saves the current command stack to that file. The commands are saved in a plain text ASCII file, one command per line.

**DOS Box** gives you a DOS session, full screen or windowed depending on how you've set the **Options** menu item **DOS In Window**. Type Exit at the DOS prompt to return to SmilerShell.

**Run** lets you choose a program from a file/directory dialog box. The filename you choose is placed on SmilerShell's command line, where you can add any needed parameters before submitting it.

**Aliases** shows you what command substitutions are currently in effect. It shows both your command-line aliases and your function-key aliases. If you don't like the way they look, there's a pushbutton to open the Settings dialog, where you can change them. You can run an aliased command from the **Aliases** list by clicking on it, or fetch its name into the main window for editing.

**Find File** brings up a dialog box that lets you search for files by name or contents. You can also bring up this same dialog box by typing 'find' on the command line, optionally with parameters giving the target filename and file contents. But to type the



command, you must first clear the command line. The advantage of the **Find File** menu item is that it lets you add a 'found' file's name to existing command line text, perhaps as a parameter to a command you are building.

**Command History** shows a list of every command you ran from the command line during this session, plus the restored list of commands you issued during the last session, plus any commands preloaded from a command-stack file at startup. This is the same dialog you get by clicking on the Command History Button. You can submit a command from here by clicking on it, or fetch it into the main window for editing. There's also a Settings pushbutton.

**Scan Directories** generates an internal list of all directories on each 'Drive To Scan For DC Data' you've listed in the Settings dialog (default is to just scan drive C). It saves this internal list to the DC info file, either a filename you specify or the default "smishell.dir" in the SmilerShell ini file's directory. The DC "/r" parameter can also generate this list on the fly when you use the DC command.

Finally, the **File** menu has an **Exit** item which terminates SmilerShell.

## The Edit Menu

The Edit menu starts with the standard Windows features **Undo**, **Cut**, **Copy**, and **Paste** for sending information to and from the Windows clipboard, and **Clear** to delete selected text in the input window.

**Remove Inactives** searches out and closes all the "(Inactive..." windows on your desktop. If you've toggled the **Options** menu item **Inactives Stay Visible** to display inactive DOS windows after their command terminates, you'll find that these windows accumulate quite rapidly. **Remove Inactives** (or its keyboard equivalent Alt+R) makes them all go away.

**Button Exceptions** is how you tell SmilerShell what to do about a single application with an unusual screen layout. Usually SmilerShell can figure out where to place its titlebar button. But if **one** of your other applications has a non-standard layout where its titlebar ought to be, the button may be placed over something important. To fix this, select **Button Exceptions**. The cursor changes to the crosshairs style. Click the mouse anywhere in the problem app's window, at the point where you'd like the button to be placed (that is, the distance from the upper left corner of the app's window). If you change your mind, press Escape to abort. This information is saved in SmilerShell's ini file from session to session. To change a **Button Exceptions** entry, select the menu item again and click on a new location in the same problem application. You can remove an entry entirely from the **Settings** dialog.

**Button Offset** moves the button left or right within the titlebar of all apps. Unlike **Button Exceptions**, setting **Button Offset** affects the button's location in **every** app, not just the one app you click on. This is handy if you have another program that also puts something in every active app's titlebar. To set this up, select the **Button Offset** item, and choose whether you want to offset from the left or the right side of the window. The cursor changes to the crosshairs style. Click the mouse in any app's window. From then on, the titlebar button will be displayed that distance from the chosen (left or right) border in every app not listed as a **Button Exception**. If you change your mind, press Escape to abort. This information is saved in SmilerShell's ini file from session to session. To change the **Button Offset**, select the menu item again and click on a new location in any application. To remove it entirely, choose **Button Offset** and click on the **Default Offset** menu item.

### Related Topics:

[The Options Menu](#)

[The Settings Dialog](#)

## The Options Menu

The **Options** menu items toggle on and off various SmilerShell features. Heavily-used features have individual menu items under the **Options** menu. Less-used or 'startup' features are listed together in the Settings dialog, at the bottom of the **Options** menu.

**Date/Time Clock** (Alt+C) puts a date/time clock in the title bar. You can set the date and time format with the **Options** menu's Settings dialog. You can also set an alarm and have it display a message when it goes off. See ALARM: Alarm Clock With Reminder Message.

**System Resources** (Alt+S) shows available system resources in the menu bar: memory, GDI, and User resources. If **System Resources** is toggled on, and the menu is toggled off, the resources report will be displayed in the command line window. Press any key and the command line comes back just as you left it. The cursor is where you left it, selections are still selected, and the key you pressed is typed into place in the command text.

**Directory** (Alt+D) show the current directory in the SmilerShell title bar.

**Overtyp**e (Alt+O) toggles the command line between insert mode and overtype mode. When it's in overtype mode a flag appears in the title bar, over the second 'e' in 'SmilerShell'.

**Topmost** (Alt+T) makes SmilerShell a topmost window, so even when inactive, it sits on top of other windows.

**Title Bar** (Alt+L) hides the title bar. This saves screen space. To move SmilerShell on the screen without a title bar, click the right mouse button in the edit area and hold it down while moving the window. Note: toggling the title bar consumes User resources on some systems.

**Menu** (Alt+M) hides the menu bar, making SmilerShell even smaller. When the menu bar is hidden, a "Show SmilerShell Menu" item is added to the System menu. Click it to get the menu bar back. Or type Alt+M from the keyboard. The keyboard accelerators (Alt+C, Alt+D, etc.) continue to work properly when the menu is hidden.

**Inactives Stay Visible** (Alt+I) controls whether, after SmilerShell runs a DOS command, the command's inactive window sticks around or goes away. Keeping those inactive windows around can be quite handy, letting you see the results of previous commands, but they do eventually clutter your screen. True, you can make them all go away using the **Remove Inactives** item on the **Edit** menu (or simply type Alt+R). But if you don't want to see them in the first place, you can simply toggle **Inactives Stay Visible** off. Or to run one command as if **Inactives Stay Visible** is set to the opposite of its current value, start that command with an **asterisk** (for example

\*dir). See [Submitting Commands](#).

**DOS In Window** (Alt+W) controls whether SmilerShell's DOS commands run fullscreen or in a window. Or to run one command as if **DOS In Window** is set to the opposite of its current value, start that command with a **right-bracket** (for example >copy foo.bat b:). See [Submitting Commands](#).

**Foreground Color** This lets you set the color of the edit control's text. Pick a standard color, or create a custom color, but note that if you pick a dithered color, Windows uses the nearest solid color instead. See [Choosing Font And Colors](#).

**Background Color** This lets you set the edit control's background color. Pick a standard color, or create a custom color. You can get a very interesting effect by choosing a dithered background color. As with **Foreground Color**, if you pick a dithered color, Windows uses the nearest solid color behind actual text characters. But it uses your chosen dithered color in the rest of the edit area. This creates an unusual raised effect as you type. Try it! See [Choosing Font And Colors](#).

**Font** You can set the command line font to any available style and size. Because of Windows limitations, italic fonts don't look as good as bold or normal fonts. See [Choosing Font And Colors](#).

**Settings** This brings up SmilerShell's [Settings](#) dialog. There are check boxes to set up startup options, titlebar button behavior, confirmation and save-state at exit, command processor usage, the date and time format in the titlebar clock, and other preferences. All your **Button Exceptions** are listed, and can be deleted if desired. You can also add, edit, or delete aliases (or you can do this with the [ALIAS and UNALIAS](#) commands), set restore time, list the drives [DC](#) should scan, and give startup filenames. See [The Settings Dialog](#) for full details.

## The Help Menu

The **Help** menu has three items: **Help**, **Popup Hints**, and **About SmilerShell**.

Use the **Help** item (or press F1) to get on-line help about SmilerShell.

Choose **Popup Hints** (or press Alt+F1) to learn interesting and useful ways of using SmilerShell.

The other item, **About SmilerShell**, is a typical About box. It gives the SmilerShell version number and contact information.

## The System Resources Menu Item

When you toggle this on, using the Options menu item **System Resources**, the **System Resources Menu Item** provides a report on three key Windows resources: bytes of available memory, GDI resources, and User resources. It changes in real time to show your currently available resources. Although it is on the menu bar, it has no menu associated with it.

If **System Resources** is toggled on and the menu is toggled off, the resources report will be displayed in the command line window. Press any key and the command line comes back just as you left it. The cursor is where you left it, selections are still selected, and the key you pressed is typed into place in the command text.

You may wonder why the **System Resources menu item** doesn't display System resources, as displayed in Program Manager's About box and other places. It turns out that "System resources" is just Windows shorthand for "the smaller of User and GDI resources." Why take up screen space with information you've already got?

## Submitting Commands

If SmilerShell is hidden, click the titlebar button with the left mouse button. The command line window will pop up and will be given the input focus. (If you right-click, the **Apps** menu will pop up next to the button, ready for you to click on an app and run it.) You can also double-click the SmilerShell startup icon at any time to reveal the command line window.

When SmilerShell has the input focus, simply type any command, just as you would at the DOS prompt. You can run Windows programs, DOS programs, or DOS internal commands like DIR or TYPE. You can use CD or CHDIR to change SmilerShell's current directory. (Or you can change directory a lot faster with the built-in SmilerShell command DC.) You can also use the built-in SmilerShell commands FIND (find files by name or contents), SHOW (display filenames and select one into the command line), ALARM (alarm clock with reminder message), ALIAS (list or change aliases), UNALIAS (remove aliases), and HISTORY (list full command history).

SmilerShell supports the 4DOS/NDOS enhanced command processor. You can use the Settings dialog to have SmilerShell handle all the 4DOS/NDOS commands that might be useful from a Windows command line. In those cases where a supported 4DOS or NDOS command is the same as a SmilerShell command (alias, history, unalias), start the command with an equals sign to pass it to the command processor unchanged (for example, =history). That is, treat it like a SmilerShell alias.

SmilerShell supports "drag and drop," so you can drop files onto its window from File Manager or any other drag-and-drop server. The filenames will be added at the end of the current command line. SmilerShell adds them in the form that takes up the least amount of space on the line. So, for example, files in the current directory will have a filename but no path.

Because SmilerShell supports File Associations, often you only have to give the filename, without naming the program to run it. SmilerShell can in these cases tell what program to run, just by looking at the filename. For example give "FILENAME.WRI" and SmilerShell knows to run Windows Write on this file. This works very nicely with SHOW or FIND: use them to pick a file, fetch the filename into the commandline, then just press Enter to run the proper program with that file.

DOS commands will run fullscreen or in a window, depending on how you have set the DOS In Window menu item.

Or you can start any command with a **right-bracket** to toggle DOS In Window for just that one command (for example >copy foo.bat b:).

If you have toggled Inactives Stay Visible to allow it, then after SmilerShell runs a submitted DOS command, the final results are displayed in an inactive window. That is, a window titled "(Inactive SOMETHING)".

When an "(Inactive..." window gets the focus as a result of running a command, SmilerShell actively takes the focus back again, so you can continue running commands from SmilerShell. Because Windows requires it, SmilerShell pauses briefly before attempting to take back the focus. By default this pause is 1000 milliseconds (1 second), but you can set it by using the **Restore Time** option in the Settings dialog. How fast can you get away with on your system?

You can use the Edit menu option Remove Inactives (or simply press Alt+R) to destroy all the "(Inactive..." windows on your desktop.

Or you can start a command with an **asterisk** (for example \*copy foo.bat b:) to toggle Inactives Stay Visible for just this one command.

Every command you submit is tested to see if it matches anything on your list of aliases. If the first word of your command matches an alias, the replacement for that alias is substituted for the first word of the command you typed. Your aliases can have parameters (%1, %2, etc, like batch files) which are substituted into them.

Or to skip alias testing for this one command, start it with a **equals sign** (for example =list). The command line will be run just as you typed it, with no alias substitution.

An easy way to remember what each of the three command line flags does is:

- \* Flash! It's gone! (or, Flash! A window appears!)
- > Moves from larger to smaller, or smaller to larger
- = Does just what it says (no alias substitution)

The three flags (\* > =) can be in any order. For example, ">\*=dir \*.exe" could be run.

You can type multiple commands on one command line. Hit Enter and they are submitted in order. Unlike most things in Windows, they are run **synchronously** -- the next command is submitted only after the previous one completes. Since the first of your multiple commands will always finish before the second one starts, you can write sequential multi-command aliases that act like batch files. They can even have parameters.

For example, you could run this from the command line:

```
d: ^ cd \dos ^ *dir *.exe /w ^ dc fle ^ >myprog param1 param2 param3
```

Or you could set up this alias:

```
foo = d: ^ cd \dos ^ *dir *.%1 /w ^ dc %2 ^ >myprog ^ param1 param2 %3
```

These commands both change drive, change directory, do a DIR, change directory



again, and run a program. The first might be typed on the fly from the command line. The second is an alias you set up ahead of time. Notice the use of % parameters. Here is how they'd work. If you gave the command **foo bat mydir otherParam** it would be turned into **d: ^ cd \dos ^ \*dir \*.bat /w ^ dc mydir ^ >myprog param1 param2 otherParam**. See [Aliases](#) for more on this.

By default these multi-commands are separated with a caret [^] but you can use the [Settings](#) dialog to select any other character you want.

The three command line flags can be used with any of the individual commands. Since each command is separate, each can have its own flags. Aliases and parameters can be used in the usual way in each individual command.

When SmilerShell finds a PIF for any program, it uses the PIF's **Inactives Stay Visible** and **DOS In Window** settings instead of its own, assuming that if you set up a PIF you did so for a reason.

By default, SmilerShell uses the command processor listed in the COMSPEC environment variable. This is usually DOS's COMMAND.COM, but some people use alternate processors such as 4DOS or NDOS. Sometimes the alternate processor has its own PIF file, for example 4DOS.COM may have a 4DOS.PIF file. This PIF will prevent SmilerShell from controlling the **Inactives Stay Visible** and **DOS In Window** settings itself, since SmilerShell will defer to the settings given in the PIF. To change this, either delete the PIF or set **UseComspec** to FALSE so SmilerShell will explicitly use COMMAND.COM, which it can control via \_default.pif.

### **Related Topics:**

[Using Arrows To Retrieve Previous Commands](#)

[Editing Commands](#)

[DOS Commands: Fullscreen Or Windowed](#)

[The Settings Dialog](#)

[DC: Directory Change The Fast Way](#)

[SHOW: An Alternative To DIR](#)

[FIND: Search For Files On Disk](#)

## Using Arrows To Retrieve Previous Commands

When you submit a command by pressing Enter, SmilerShell stores it internally in a command stack. To retrieve a command, press the up/down arrow keys until the command you seek is displayed. Press the up-arrow to see the previous command, or the down-arrow to see the next command.

You can search for a particular previous command to be displayed. Let's say you want to find the command "dir \windows\system\\*.ini /p" that you ran some time before. Just type D before you press the up-arrow. SmilerShell will find the most recent command that started with D. You are not limited to just the first letter; you can type as much of the previous command as you need to specify the match you want. If the first match isn't the command you are looking for, press that arrow key again until the command you want comes up. The same match-string is used until you type something that changes a displayed command. Matches are not case-sensitive. The command-line flags (\* > =) are ignored when checking for a match.

To simply retrieve all commands in order, just make sure the command line is blank when you first press the arrow key. You can clear the command line by pressing Escape.

You can also click the [Command History](#) button, or give the command [HISTORY](#), or use the [File](#) menu's **List Commands** item to gain access to your entire command history. This includes the restored list of commands you issued during the last session, plus any commands preloaded from a command-stack file at startup

### **Related Topics:**

[Submitting Commands](#)

[Editing Commands](#)

## Editing Commands

The normal editing keys allow you to move within the command line. Use Home, End, left-arrow and right-arrow to move within the command line. Ctrl+left-arrow move one word to the left, and Ctrl+right-arrow move one word to the right. You can clear the command line by pressing Escape.

SmilerShell's command line can be in either insert mode or overtype mode. Toggle this with the Options menu item **Overtyp**e, or just type Alt+O. In overtype mode, a flag appears in the title bar, above the second 'e' in 'SmilerShell.'

### **Related Topics:**

[Using Arrows To Retrieve Previous Commands](#)

[Submitting Commands](#)

[The Options Menu](#)

## Size Of Window

SmilerShell will accept commands of up to 128 characters (the DOS command line limit). You can make the command line window as wide as you like. However, there is never any need to make it more than one line high! If you try, it snaps back.

When maximized, SmilerShell takes up only the top line of your screen. You can set up a very useful configuration by setting SmilerShell as a topmost window, then maximizing it.

For a smaller SmilerShell window, use the Options menu to toggle off the menu and title bar, then mouse the window as small as you like. You can make it as small as an icon, or even smaller.

Of course for the smallest window of all, click the Hide! item in SmilerShell's menu. The command line will vanish, and take up no screen space at all. To get it back, left-click the SmilerShell button that hops into the title bar of whichever application you've currently got activated.

### **Related Topics:**

[The Hide! Menu Item](#)

## Getting Rid Of Inactive Windows

The Options menu item **Inactives Stay Visible** controls whether, after SmilerShell runs a DOS command, the command's inactive window sticks around or goes away. Use the Options menu to set this to your preference. By default SmilerShell sets the menu toggle to match the current systemwide value found in the "\_default.pif" file in your Windows directory.

If you've toggled **Inactives Stay Visible** to allow it, each DOS command ends by firing up its own "(Inactive..." window. This is handy, letting you see the results of previous commands, but it does eventually clutter your screen. To make them all go away, use the **Remove Inactives** item on the Edit menu, or simply type Alt+R.

To toggle **Inactives Stay Visible** for just one command, start the command line with an **asterisk** (\*list foo.txt for example). You can use this with the **equals sign** or **right-bracket** flags if you want to (for example =\*>dir).

## About Internal DOS Commands

SmilerShell runs most DOS commands in a subshell. For internal DOS commands that affect the working environment, this is tricky. The subshell starts up with a copy of the parent's environment, things like current directory, environment variables, settable DOS version, etc. If you alter an environment variable or change directory in a subshell, the parent shell's information does not change. SmilerShell can support all the internal DOS (and 4DOS/NDOS) commands you're likely to want. It supports CD/CHDIR and PATH itself (see below), and for the rest calls the system's command processor. In addition, there are three "semi-supported" internal DOS commands: CHCP, SET, and VER. In DOS, these can both set and show environment values. If you enter one of these, SmilerShell will show their current value. However, because you are in a subshell, not your actual environment, you can't change these values in your actual working area through SmilerShell (or through any Windows program, generally).

SmilerShell handles CD/CHDIR and PATH by itself. For CD/CHDIR it changes the current directory on the specified drive, or its own current directory if no drive letter is given. CD or CHDIR with no parameter shows the current directory, handy when the title bar is toggled off.

For PATH, the behavior depends on the command's parameters. PATH by itself shows the current path in SmilerShell's edit window. PATH=TEXT sets SmilerShell's copy of the path to the TEXT you specified. PATH= (with nothing after the equals sign) sets SmilerShell's path to be the same as the current systemwide path. For more information on this, see [PATH: Change The Search Path](#).

If you use the 4DOS (or NDOS, same thing) command processor, you can use the **Settings** dialog to set up SmilerShell to correctly handle 4DOS/NDOS internal commands. For more information on this, see [Using 4DOS/NDOS Internal Commands](#).

To summarize:

Supported Internal DOS Commands: CD, CHDIR, COPY, DATE, DEL, DIR, ERASE, MD, MKDIR, PATH, REN, RENAME, RD, RMDIR, TIME, TYPE, VOL

"Semi-supported" Internal DOS Commands: CHCP, SET, VER

Unsupported Internal DOS Commands: CLS, CTTY, EXIT, PROMPT, VERIFY and the batch file commands.

Supported Internal 4DOS/NDOS Commands: ?, ALIAS, BEEP, CDD, DESCRIBE, DIRS, ESET, FFIND, FREE, GLOBAL, HISTORY, LIST, LOADBTM, LOG, MEMORY, MOVE, POPD, PUSHD, REBOOT, SELECT, SETDOS, SWAPPING, TEE, TIMER, TRUENAME, UNALIAS, Y



## Aliases

An **alias** is a short command that is replaced with a longer command. Some people call them macros. There are two kinds of aliases in SmilerShell. In the first kind of alias, you type a (typically, short) command line and press Enter, and the first word of the line is replaced with another (typically, long) string. The rest of the original command line is tacked on after the replacement string. You can define up to about 100 of these **type-in aliases**. In the second kind, you press a function key and a predefined command is submitted. You can define one of these **function-key aliases** for each of F2 through F12 (F1 is reserved for Help).

**Type-in Aliases:** Let's look at the first kind. Say you set up the alias:

```
dirprog=dir c:\develop\source\*.*
```

Whenever you enter the command "dirprog", SmilerShell will replace it with, and actually submit, the command "dir c:\develop\source\\*.\*" to be run. This saves wear and tear on your typing fingers.

You can put parameters on this kind of alias. In our example, you could enter

```
dirprog /o /p
```

and SmilerShell would run the command

```
dir c:\develop\source\*.* /o /p
```

by adding the original parameters after the substituted alias.

A typed alias is used just like any other command; type it in (with parameters if any) and press Enter. SmilerShell looks at the first word on each command line to see if it's an alias.

To avoid alias checking for a particular command, start it with an equals sign. For example, if you actually had a program called "dirprog" that you wanted to run instead of the alias defined above, you could submit this:

```
=dirprog
```

Because the command line starts with an **equals sign**, SmilerShell skips the alias testing for this command.

You can use the **equals sign** flag with the **right-bracket** or **asterisk** flags if you want to. In this example, you could type **\*>=dirprog** and press Enter.

SmilerShell allows multiple commands on one line. You can alias such multi-



command lines. For example:

```
bigcmd = cd \dos ^ *dir *.exe /w ^ dc file ^ >=myprog param1 param2 param3
```

Notice the use of **equals**, **right-bracket**, and **asterisk** flags on the multi-command line. Since each command is separate, each can have its own flags.

Type-in aliases can have runtime parameters %0 through %9, similar to DOS batch file parameters. %0 is the alias-part itself, and %1-9 are the first nine arguments. Here's a simple example that changes a file's extension from ASC to anything else:

```
newextn=ren %1.asc %1.%2
```

To change filename.asc to filename.txt you'd use it like this:

```
newextn filename txt
```

Runtime parameters really shine in multi-command aliases. For example, to implement your own "move" (copy and delete) command you could do this:

```
move=copy %1 %2 ^ del %1
```

Now, to move a file or group of files, you could give the command "move \*.exe b:" and SmilerShell would copy the files, then delete the originals.

Any command line arguments that are not used up by % parameters are left at the end of the command line. So "move \*.exe b: fee foo" would turn into "copy \*.exe b: ^ del \*.exe fee foo".

How about an alias like "move=copy %1 %3 ^ del %1" that doesn't use an argument? If you wrote an alias to use %1 and %3 but never refer to %2, the first and third argument would be used in the command, and the second argument would be put at the end of the command line.

**Function Key Aliases:** The second kind of alias is where you attach a command to a function key. Just press the function key and the command is submitted. You don't need to press Enter to submit it. Function keys F2 through F12 can be set up this way.

For example, let's say you have set up the alias:

```
(F5)=copy c:\develop\source\*. * b:\
```

Now, whenever you press F5 in SmilerShell, the command "copy c:\develop\source\\*. \* b:\" will be submitted. It's very handy, no need to press Enter.

Function key aliases can be a multi-command line (see above) but they cannot have parameters.

**In General:** You can define or change aliases from the command line one at a time with the ALIAS and UNALIAS commands. Or use the Aliases section of the Settings dialog to work with them all at once. In the Settings dialog, list aliases one per line, in the form alias=replacement. The left side of a **function-key alias** has the key name in parentheses, as in the example above. The left side of a **type-in aliases** can be whatever you like, as long as the alias-part has no embedded spaces. The replacement-part can be whatever you like.

By default, alias testing is case-sensitive. You can change this in the Aliases section of the Settings dialog.

Aliases can reference other aliases, but be careful to avoid self-referencing infinite loops (alias 1 defined in terms of alias 2 which is defined in terms of alias 1 which is ...).

**Related Topics:**

The Settings Dialog

ALIAS and UNALIAS: Create, Change, Remove Aliases

## DC: Directory Change The Fast Way

**DC** (Directory Change) is a built-in alias that lets you change directory very quickly. Instead of having to type in the entire pathname, you only need to give it the first few letters of the endpoint (leaf-node) directory you want. For example, instead of typing "cd \c8\mfc\samples\fileview" you could type "dc fi" and press Enter. If "fi" is enough to unambiguously specify one directory, **DC** takes you right there. If what you typed is ambiguous (maybe there's more than one directory whose name starts with "fi") a window pops up, showing all your possible matches in alphabetical order. The first possible match is highlighted. If there was no possible match, nothing is highlighted. Double-click on your choice, or single-click and press OK.

At the top of the box is the number of directories **DC** knows about. There's a button to re-scan the directory list as well.

To re-scan the list and change directory all at once, use the /r parameter. For example "dc /r fi" would first re-scan, and only then look for that fi directory (in the new list) as described above. You can use a dash ("-r" instead of "/r") if you prefer.

If the endpoint directory is on a different drive, **DC** will first change drives, then change to the desired directory. There's no need for you to manually change drives first. **DC** does it for you.

By default, the **DC** data is stored in the file "smishell.dir" in the same directory as the initialization file (you can specify another name and location for this file in the Settings dialog). SmilerShell creates this file the first time you use **DC** (with your permission), or whenever you use the File menu item **Scan Directories**. It contains the name of every directory on each drive that was scanned. These are the directories that **DC** can change to. To indicate what drives you want scanned, set **Drives To Scan For DC Data** in the Settings dialog. For example, if this is in Settings:

**Drives To Scan For DC Data** [cdm ]

then SmilerShell will generate a list of all directories on your c, d, and m drives. (The list of drives can be separated by commas or spaces [ c, d, m ], or bunched together as in the example above.)

If the directory layout of your system is constantly in flux, you may want to check the Settings box **Startup DC Scan** so SmilerShell rescans your **DC** information every time it starts.

Maybe you have some other program called **DC** that you'd like to run? Since SmilerShell's **DC** acts like an alias, you can bypass it by starting the command line with an equals sign.

**Related Topics:**

## The Settings Dialog

## Command Stack Files

If you have a set of commands you'd like to be able to load into SmilerShell, create a command stack file. This is simply a plain-text file with one command per line. By default, the command stack file name is "smishell.stk" and it is in the same directory as the ini file. If this file exists, its commands are preloaded.

However, you can use any filename, location, or extension you like. Give your preferred filename in the Settings dialog.

Command stack files can also be loaded or saved at any time from the File menu.

A preloaded command-stack is read before the command history from the most recent SmilerShell session. That is, the previous-session commands will look more "recent" in the history list than command-stack commands.

## The Initialization File

The **Settings** dialog, on the **Options** menu, displays the current values of most SmilerShell initialization parameters. (All the other values are also readily available, shown as menu item checkmarks, button offset, etc). SmilerShell initializes itself at startup by reading these parameters from an initialization file. This is a plain-text file, so you could use Notepad or another ASCII text editor to edit it by hand. However, SmilerShell's Settings dialog and menu toggles provides a much more convenient way to do this. They let you manipulate every option, with as much flexibility as if you were editing your ini file by hand, but in a much more convenient manner. Most people will choose to do it this way, but for completeness all ini file settings are documented below.

If you started SmilerShell without an initialization file, you'll be asked if you'd like to create one, filled with reasonable defaults, before proceeding. When convenient you can change these values with the Settings dialog and menu toggles.

If you want SmilerShell's initialization file to be named something other than "smishell.ini" or be somewhere other than the same directory as the SmilerShell program, give that information on the 'SmilerShell icon's 'Properties' command line. To do this, use Program Manager's Properties dialog (it's under the File menu). Add a space after "smishell.exe", then the flag **/ini=** (lower case) and the drive and directory in which to find the ini file, with no embedded spaces, as in the following example:

Command Line: **c:\smishell\smishell.exe /ini=c:\dir1\subdir2\myfile.ext**

If you don't specify otherwise, SmilerShell looks for its **DC** information file and any startup command stack file in the same directory as the initialization file.

SmilerShell's initialization file is set up just like every other ini file in Windows. For each section, there's a section header in brackets, under which are entries that are set to values:

```
[SmilerShell Params]
AliasesCaseSens=TRUE
BGcolor=20,200,150
buttonoffset=75
cmdfile=smishell.stk
cmdstack=dir \dos;type \autoexec.bat;copy \config.sys junk.tmp;
confirmexit=TRUE
dateformat=M
dirfile=c:\smishell\smishell.dir
DOSinWindow=FALSE
FGcolor=255,0,0
Font=-19,0,0,0,700,0,0,0,0,3,2,1,34,Arial
overtyping=FALSE
```

multicmdchar=^  
prevdir=<set by SmilerShell when exiting>  
prevposition=<set by SmilerShell when exiting>  
restoretime=400  
scandrives=bcde  
showbutton=TRUE  
showclock=FALSE  
showdir=FALSE  
showinactives=TRUE  
showmenu=TRUE  
showresources=TRUE  
starthidden=TRUE  
startupDCscan=FALSE  
timeformat=12  
titlebar=TRUE  
topmost=TRUE  
Use4DOS=FALSE  
UseComspec=TRUE  
winwidth=500 or PREV

[SmilerShell Aliases]

TypedAlias=Replacement  
(F3)=Replacement  
word = c:\winword\winword  
(f2) = type \autoexec.bat  
(F5)= sol  
dirprog=dir c:\mydir\mysubdir\programs\\*.exe  
(F12)= notepad \config.sys

[SmilerShell Button Exceptions]

WINWORD=24,18  
ZoomIn=11,82

[SmilerShell Apps]

1=dir \*.txt,DIR Text Files,TRUE  
2=chkdsk,Check Disk,FALSE

[SmilerShell CmdHist]

1=dir /w  
2=winword myletter.doc  
3=notepad \autoexec.bat  
NumCmds=3

You don't have to have all of the parameters. They don't have to be in the order shown above. If you don't specify a parameter, the SmilerShell default is used for it. If you do specify a parameter, you can comment it out by putting a semicolon at the

beginning of its line.

**[SmilerShell Params]:** Parameters can be changed at any time from the Settings dialog or menu toggles. The defaults, and the parameter meanings, are:

**AliasesCaseSens=TRUE** Each command you submit is tested to see if it matches a string on your alias list. By default, this alias testing is case-sensitive. To change this, set AliasesCaseSens=FALSE.

**BGcolor=255,255,255** The RGB value of the command line's background color. By default the background is white. The starter ini file that SmilerShell generates for you set this to blue.

**buttonoffset=0** Usually SmilerShell's button is snug against the titlebar's left edge. Use **buttonoffset** to move it to the right in apps not listed as **Button Exceptions**. Or enter the **buttonoffset** parameter automatically by using the **Button Offset** item on the Edit menu.

**cmdfile=** File from which to pre-load commands. By default, no filename specified, so no commands are pre-loaded.

**confirmexit=TRUE** If this is TRUE (the default), you'll be asked to confirm that you really do want to exit from SmilerShell.

**dirfile=** File from which to load the DC information. SmilerShell creates this file the first time you use DC (with your permission), and whenever you **Scan Directories**. The default name is "smishell.dir" and the default location is the same directory as the SmilerShell ini file. Use **dirfile=** to give it a non-default name or location. A filename without a path is assumed to be in the same directory as the SmilerShell ini file.

**dateformat=M** This parameter governs the date format used in the titlebar clock. It can be set to **M** (use mm/dd/yy format), **D** (use dd/mm/yy format), or **Y** (use yy/mm/dd format).

**DOSinWindow=TRUE or FALSE; default is whatever the systemwide "\_default.pif" flag is set to.** Do you prefer to run DOS commands fullscreen or in a desktop window? If you don't specify, the current systemwide value, found in the file "\_default.pif" in your Windows directory, is the initial setting. If you have set **winwidth** to PREV, indicating that you want SmilerShell to come up next time with the same location and status as it ended this time, the DOSinWindow parameter will be updated when SmilerShell exits. This parameter governs the initial setting of the DOS In Window item on the Options menu. It is temporarily toggled when you use the **right-bracket** command line flag.

**FGcolor=0,0,0** The RGB value of the command line's foreground (text) color. By default the foreground is black.



**Font=-13,0,0,0,700,0,0,0,0,1,2,1,34,System** By default, SmilerShell uses the System font in its command line with these font parameters.

**multicmdchar=^** You can put more than one command on a line. By default, SmilerShell separates multiple commands with a caret [^]. Use this parameter to set the separator character to something else.

**overtypemode=FALSE** By default, SmilerShell's command line starts in insert mode, not overtype mode. If you have set **winwidth** to PREV, indicating that you want SmilerShell to come up next time with the same location and status as it ended this time, the **overtypemode** parameter will be updated when SmilerShell exits.

**prevdir=<previous drive and directory>** If you have set **winwidth** to PREV, indicating that you want SmilerShell to come up next time with the same location and status as it ended this time, SmilerShell creates the **prevdir** parameter and saves its current directory there at exit. Next time, SmilerShell starts up in this directory. If you've set **winwidth** to PREV but there's no **prevdir** parameter at startup, SmilerShell starts up in the program directory.

**prevposition=number, number, number, number** If you have set **winwidth** to PREV, indicating that you want SmilerShell to come up next time with the same location and status as it ended this time, SmilerShell creates the **prevposition** parameter and saves its current location there. The four numbers saved are the x, y, height, and width in screen coordinates. If you've set **winwidth** to PREV but there's no **prevposition** parameter at startup, SmilerShell chooses its location as if **winwidth** wasn't specified.

**restoretime=1000** How many milliseconds to pause before trying to regain the focus from an "(Inactive..." window. Windows needs a little pause here. How little can your system get away with? Default is one second (1000 milliseconds).

**scandrive=c** By default, when you **Scan Directories** only the C drive is scanned for directory names to be used with SmilerShell's DC command. DC can change to endpoint directories on other drives. Here is where you tell it what drives you want it to be aware of.

**showbutton=TRUE** By default, the SmilerShell button hops into the titlebar of whichever application is currently active. To disable this feature, set **showbutton** to FALSE.

**showclock=FALSE** By default, the date and time clock on the title bar is not displayed. If you have set **winwidth** to PREV, indicating that you want SmilerShell to come up next time with the same location and status as it ended this time, the **showclock** parameter will be updated when SmilerShell exits. By default, SmilerShell's clock uses a 12-hour (am/pm) time format, and a US-standard

MM/DD/YY date format. To use other formats, set `timeformat` and/or `dateformat`.

**showdir=TRUE** By default, the current directory is displayed as part of the SmilerShell window's title. If you have set **winwidth** to `PREV`, indicating that you want SmilerShell to come up next time with the same location and status as it ended this time, the **showdir** parameter will be updated when SmilerShell exits.

**showinactives=default is TRUE or FALSE, whatever the systemwide "\_default.pif" flag is set to.** After a DOS command ends, do you want to see the inactive results window or just have it vanish? If you don't specify a preference, the current systemwide value, found in the file `"_default.pif"` in your Windows directory, is the initial setting. If you have set **winwidth** to `PREV`, indicating that you want SmilerShell to come up next time with the same location and status as it ended this time, the **showinactives** parameter will be updated when SmilerShell exits. This parameter governs the initial setting of the **Inactives Stay Visible** item on the **Options** menu. It is temporarily toggled when you use the **asterisk** command line flag.

**showmenu=TRUE** By default, the menu bar is displayed. If you've set **winwidth** to `PREV`, indicating that you want SmilerShell to come up next time with the same location and status as it ended this time, the **showmenu** parameter will be updated when SmilerShell exits.

**showresources=FALSE** By default, resources are not displayed. If you have set **winwidth** to `PREV`, indicating that you want SmilerShell to come up next time with the same location and status as it ended this time, the **showresources** parameter will be updated when SmilerShell exits.

**starthidden=TRUE** By default, after SmilerShell puts its button into the active application's title bar, it hides itself. To prevent it from vanishing, set **starthidden** to `FALSE`.

**startupDCscan=FALSE** By default, SmilerShell only scans when you tell it to, either by choosing the **Scan Directories** menu item or with **DC's "/r"** flag. If the directory layout of your system is constantly in flux, you may want to set the ini file parameter **startupDCscan** to `TRUE` so SmilerShell rescans your **DC** information every time you start

**timeformat=12** By default, SmilerShell's clock uses a 12-hour (am/pm) format. To use a 24-hour format, set `timeformat=24`.

**titlebar=TRUE** By default, the title bar is displayed. If you have set **winwidth** to `PREV`, indicating that you want SmilerShell to come up next time with the same location and status as it ended this time, the **titlebar** parameter will be updated when SmilerShell exits.

**topmost=FALSE** By default, SmilerShell is not always on top of other windows. If

you have set **winwidth** to PREV, indicating that you want SmilerShell to come up next time with the same location and status as it ended this time, the **topmost** parameter will be updated when SmilerShell exits.

**Use4DOS=FALSE** By default, SmilerShell does not know to recognize 4DOS/NDOS commands as valid, and send them to the system command processor.

**UseComspec=TRUE** By default, SmilerShell uses the command processor listed in the COMSPEC environment variable. This is usually DOS's COMMAND.COM, but some people use alternate processors such as 4DOS or NDOS. To have SmilerShell explicitly use COMMAND.COM, set **UseComspec** to FALSE.

**winwidth=number or PREV** The initial width of the SmilerShell window, in Windows device units. If you don't specify a **winwidth**, SmilerShell sets it wide enough to show the menu items. This works out to about 600 Windows device units if **showresources** and **showapps** are TRUE, 300 if both are FALSE. You can also set **winwidth** to PREV, indicating that you want SmilerShell to come up next time with the same location and status as it ended this time. If you have, then at exit the system updates a number of parameters and saves its current location and directory.

**[SmilerShell Aliases]** are specified one per line, in the form:

alias=replacement

The alias-part of a type-in alias is a single word, with no embedded spaces. Function-key aliases have the key name (F2 through F12) in parentheses.

You can define up to about 100 type-in aliases, depending on their length. You can define one function-key aliases for each of the keys F2 through F12 (F1 is reserved for Help).

For either kind of alias, the replacement-part can be any number of words, anything you can type on one line, up to the DOS limit of 128 characters per submitted command.

In the replacement-part, the runtime parameters %0 through %9 will be replaced with the matching command line arguments. %0 is the alias-part, and %1-9 are the first nine arguments.

The replacement-part can be made up of multiple commands. They are separated by the **multicmdchar** (^ by default) as on the command line. Parameters are replaced before the multi-command line is broken into individual commands. Any arguments not used by % parameters are put at the end of the multi-command line.

By default, aliases are case-sensitive. You can change this by setting

**AliasesCaseSens=FALSE.** Aliases can reference other aliases, but be careful to avoid self-referencing infinite loops (alias 1 defined in terms of alias 2 which is defined in terms of alias 1 which is ...).

**[SmilerShell Button Exceptions]** If you have non-standard Windows applications with unusual titlebar setups, you can use the **Button Exceptions** menu item to tell SmilerShell where in those windows you'd prefer the button to go. Your preferred button locations are saved in the [SmilerShell Button Exceptions] section, one application per line, in this format:

AppWindowID=x,y

To change an application's button-placement entry, use the **Button Exceptions** menu item to select a new button location for that app. When you do, SmilerShell will edit the correct ini file entry automatically.

Note that **Button Exceptions** affect only one application at a time. To move the button in every application's titlebar, consider using **Button Offset**.

**[SmilerShell Apps]** This is the list of up to 199 "favorite applications" that is displayed when you select the **Apps** top-level menu item, or right-click SmilerShell's titlebar button. You should never need to edit this section by hand. It's much easier to use the **Apps** menu item **Edit Apps List**. The apps are in the form:

N=Command, MenuLabel, RunNow

**N** is a number; the commands are numbered 1, 2, 3, 4, etc. **Command** is the actual command to submit; it can use aliases, command-line flags (\*>=), be a multi-line command, etc. **MenuLabel** is the text displayed on the menu. **RunNow** controls what to do when the item is selected from the menu. It is either TRUE to submit the **Command** immediately, or FALSE to select the **Command** into the commandline for editing.

**[SmilerShell CmdHist]:** If you've checked Save State in the Settings dialog, SmilerShell will save all the commands you issued during this session. They will be restored next time as part of the command history, after any startup command stack file (that is, they will look more "recent" in the history list). This section is where these commands are saved from session to session. They are saved one per line, in the numeric order in which they were issued, in the form:

N=Command

**N** is a number; the commands are numbered 1, 2, 3, 4, etc. **Command** is the actual

command that was submitted. This section also has a **NumCmds** entry, indicating how many commands are listed in the section.

**Related Topics:**

[Aliases](#)

[Command Stack Files](#)

[The Apps Menu](#)

## Why Is This A Shell?

The word **shell** is sometimes used for a wrapper that surrounds other applications and hides them. SmilerShell does that. You can set up its Apps menu to run anything you want, quickly and easily, from anywhere. No muss, no fuss.

But SmilerShell also does the opposite of that. SmilerShell's commandline lets you "shell out," making all the power of the command line available from an environment in which that power is not otherwise accessible.

One word, two meanings. Faster, easier, more powerful. That's what SmilerShell is, and that's why it's a shell.

## Notices

VERSION: SmilerShell version 3.1

SYSTEM REQUIREMENTS: Requires Microsoft Windows 3.1

SOFTWARE LICENSE: Anyone is welcome to distribute unregistered evaluation copies of SmilerShell, in its entirety as distributed with this file, subject to these conditions:

- 1) None of the files in this package may be modified or deleted.
- 2) Vendors or distributors who generally pay royalties must notify the author that they are distributing SmilerShell.
- 3) Vendors or distributors must stop distributing SmilerShell if asked to do so by the author or author's representative.

In addition, the attached VENDINFO data record is hereby incorporated by reference. Any distribution satisfying all the distribution requirements expressed in that data record is hereby authorized.

After purchasing, copies of SmilerShell may not be distributed. Only one user is authorized to use the program, on one computer. It may not be used in a multi-user setting without first obtaining a site license. It may be duplicated only for the purpose of making a reasonable number of backup copies.

DISCLAIMER: The author of this software package, Barry Smiler, has used his best efforts in producing this software and documentation. These efforts include the research, development, and testing of the software, and production of the documentation.

WARRANTY: The author makes no warranty of any kind, expressed or implied, with regards to the software or the documentation. The author shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of this software package.

COPYRIGHT: All SmilerShell software and documentation copyright 1993,1995 Barry Smiler.

CONTACTING THE AUTHOR: Barry Smiler, the author of SmilerShell, can be contacted through CompuServe email (72340,375), Internet email (72340.375@compuserve.com), U.S. mail (Bardon Data Systems, 1023 Key Route Blvd., Albany CA 94706), or phone (510-526-8470).

## Uninstalling SmilerShell

SmilerShell tries to be considerate of the rest of your Windows system. Other than putting Microsoft's **ctl3dv2.dll** in your Windows system directory if not already there (many programs use this Microsoft systemwide tool), it just copies its own files to the directory you specify, and adds its icons to the Program Manager group you specify. If you decide to uninstall, just delete the specified files and icons.

If you have put the SmilerShell ini file in another directory, delete it. If you have used SmilerShell's DC command, delete the DC info file. By default it is named "smishell.dir", and is in the same directory as SmilerShell's ini file.

### **Related Topics:**

[Installing SmilerShell](#)



## Installing SmilerShell

SmilerShell includes the following files which are installed to the directory you choose:

**smishell.exe** the program  
**smishell.hlp** the documentation, in Windows help file format  
**readme.txt** overview and installation instructions  
**sample.stk** sample command stack file  
**whatsnew.txt** new features and revision history  
**orderfrm.wri** SmilerShell's order form / invoice  
**dealers.txt** toll-free numbers and SmilerShell vendors worldwide

This file is installed to the Windows system directory if not there already:

**ctl3dv2.dll** Microsoft-supplied system file to give SmilerShell a 3-D look

SmilerShell also comes with these files:

**install.exe** automated SmilerShell installer  
**file\_id.diz** formatted description file, for BBS uploads  
**vendinfo.diz** formatted description file for software vendors  
**vendor.txt** plain-text description file for software vendors

You can install SmilerShell automatically, using the enclosed auto-installer. To do this, simply run **install.exe** from within Windows. You can run it using File Manager, or the Run item on Program Manager's File menu, or in whatever other convenient way you choose. Give it the directory to put SmilerShell's files into, and the Program Manager group name for the SmilerShell icons (appropriate defaults are suggested). It'll do the rest. The installer will make no changes to your system set-up or to any systemwide files. It copies the SmilerShell files to the one directory you specify, and puts its icons in the one Program Manager group you choose. The only exception is **ctl3dv2.dll**, a Microsoft-supplied enhancement to Windows that gives programs a 3-D look. This file is installed into the Windows system directory if you don't already have it.

Optionally, you can set up an initialization file. But if you start SmilerShell without an initialization file, you'll be asked if you'd like SmilerShell to create one and fill it with reasonable values. You can view and change these values with the Settings dialog.

Optionally, give a non-default location for the initialization file. Perhaps you're on a network, and want to install SmilerShell on a server, but let everyone have their own local setup. See The Initialization File.

Optionally, you can create a command stack file, having a list of commands that you want loaded into SmilerShell at startup. See Command Stack Files.

**Related Topics:**

[Uninstalling SmilerShell](#)

## Quick Start / Hints And Tricks

Windows is great, but sometimes it can be awkward. Getting to the icon to launch an application isn't always easy. Finding files is difficult at best. There's no good way to keep an eye on important systemwide resources. And even when you have all the pieces, sometimes it'd be faster to just type a command -- but how?

That's what SmilerShell is for.

SmilerShell is a compact yet powerful Windows control center that takes NO space on your desktop. Until you need it, SmilerShell is just a tiny button that hops into the titlebar of whichever app you're working in. Right-click the button to see your configurable Apps list and launch a new program, or switch to a currently-running task. Or left-click the button to reveal the ultimate Windows command line, which supports pipes, redirection, and internal DOS commands (and of course runs Windows programs too). SmilerShell has the best command line you've ever seen, as if the plain-vanilla DOS prompt was enhanced by lots of handy utilities.

Here's how to get the most out of SmilerShell.

**INSTANT INSTALL:** You can install SmilerShell automatically, using the enclosed installer. Simply run install.exe from Windows using File Manager or the Run item on Program Manager's File menu, or in whatever other convenient way you choose. Give it the directory to put SmilerShell's files into, and the Program Manager group name for the SmilerShell icons (defaults are suggested). It'll do the rest. The installer will make no changes to system or initialization files. Other than putting Microsoft's ctl3dv2.dll in your Windows system directory if not already there (many programs use this Microsoft systemwide tool), it just copies its own files to the directory you specify, and adds its icons to the Program Manager group you specify. If you decide to uninstall, just delete the specified files and icons.

**PRESS THE BUTTON:** SmilerShell's activation button hops into the title bar of whichever application is currently active. Right-click the button and the Apps menu appears, ready for you to launch one of your listed apps or switch to a currently-running program. Or left-click the button to reveal the command line window. Then later, hit the SmilerShell menubar's "Hide!" item and the commandline window vanishes again.

**BUTTON EXCEPTIONS:** Have a non-standard Windows application with an unusual titlebar setup? Use the Button Exceptions menu item to tell SmilerShell where in that window you'd prefer the button to go.

**BUTTON OFFSET:** If you have another utility that uses every active app's titlebar, tell SmilerShell how to move its button out of the way by setting up a Button Offset.

**FAVORITE APPLICATIONS:** List your favorite applications on the Apps menu. Then

just click on one to either select its name into the command line, or run it straight off (you can set it up either way). The Apps menu also lists all currently-running programs. Click on one to switch to it.

**BUILD YOUR OWN COMMANDS:** You can list all sorts of things on the Apps menu. All Windows or DOS commands and programs, of course. But also SmilerShell aliases, multiple commands on one line, DOS internal commands, pipes and redirection. Your commands can have parameters, too. Want to have a special command to start your spreadsheet program, pre-loaded with specific data, from anywhere in your system, with one click? Here's how to do it!

**COMMAND HISTORY AND SEARCH:** Every time you run a command from the commandline, it is saved. To find a previous command of interest, type the first letter or two of that command, then press the up-arrow (search back) or down-arrow (search forward) key. The last command is connected to the first, so you can search in either direction. Arrows on a blank line show all commands in order. If you've checked **Save State** in the Settings dialog, all the commands you gave this time will be in the command history when you start next time, handy for searching. To see the full command history list, click the Command History button, or type HISTORY on the commandline. The full history list is also available from the File menu.

**COMMAND LINE EDITOR:** A retrieved previous command, or anything else you type, can be edited to suit. Think of SmilerShell as a one-line word processor. It supports insert mode, overtyping mode, and clipboard cut/paste.

**ALIASES:** When you press Enter, the first word of the command is compared to the alias list. If it matches, the alias is substituted for that first word. You can skip the alias testing by starting the command with an equals sign. You can also hang aliases off the function keys F2 through F12; hit the key and the command runs. Both kinds of aliases are set up in the Options menu's Settings dialog, or from the commandline with the ALIAS and UNALIAS commands. A full alias list is also available on the File menu.

**RUNTIME PARAMETERS IN ALIASES:** Runtime parameters (%1, %2, etc.) make it easier to tell aliases what to do when you run them. And if you alias multiple commands on one line, the alias acts almost like a batch file, all within SmilerShell!

**PERSONALIZE YOUR SHELL:** You can set the command line's font and colors any way you like.

**HELP:** Of course there's full Windows Help. But in addition, you can use the Popup Hints (Alt+F1) to get tips on interesting ways to use SmilerShell. Popup Hints are especially handy for new users.

**ALTERNATE COMMAND PROCESSORS:** Do you use 4DOS or NDOS? Tell SmilerShell to recognize 4DOS/NDOS commands, and set SmilerShell to use the

4DOS/NDOS command processor instead of plain old COMMAND.COM. It's in the Settings dialog.

**GET SMALL:** SmilerShell has a very small window, but you can make it even smaller. Use the Options menu to get rid of the menu and title bar. Or type Alt+M to toggle the menu, Alt+L to toggle the title bar. Then mouse SmilerShell as small as you like. It'll go smaller than an icon!

**QUICK DIRECTORY CHANGE:** Type DC and the first few letters of the directory you want to be in. If it's unambiguous, boom, you're there, otherwise a list box pops up with the first possible match highlighted. If you haven't used DC yet, you'll be asked for permission to scan the drives listed in the Options menu's Settings dialog. If you scan more than one drive, DC can change drive as well as directory to get you where you want to go.

**SMILERSHELL NEVER FORGETS:** Check a box in the Settings dialog and SmilerShell will start up next time in the same directory, same screen position, and with the same settings, as when you shut it down this time.

**DOS IN A WINDOW:** Do you prefer to have DOS commands run fullscreen or in a window? Toggle this on the fly with the DOS In Window menu item. Or to run one command as if DOS In Window is set to the opposite of its current value, start that command with a right-bracket (for example >dir).

**INACTIVES STAY VISIBLE:** After you run a DOS command, do you want the command's inactive window to stick around, or immediately vanish? Toggle this flag, called Inactives Stay Visible, from the Options menu. Or to run one command as if Inactives Stay Visible is set to the opposite of its current value, start that command with an asterisk (for example \*dir).

**REMOVE INACTIVES:** Too many inactive windows cluttering your screen? Get rid of 'em with this Edit menu item, or just type Alt+R from the keyboard.

**DATE AND TIME:** Toggle the titlebar date/time clock from the Options menu, or just type Alt+C. Prefer 12-hour or 24-hour time? Various date formats? Set it the way you like it with the Options menu's Settings dialog.

**ALARM CLOCK AND REMINDER:** Use the built-in ALARM command to set the alarm clock. You can even give it a message to display when the alarm goes off.

**WORK WITH FILES:** The built-in command SHOW is often a useful alternative to DIR, since SHOW's file list lets you click on a filename to select it into the command line.

**FIND FILES:** Use the FindFile dialog to search for files by name or contents, anywhere in your system. It accepts wildcards and can look for files containing specific data. When you find the file you want, you can go to its directory or fetch its full name into

the commandline. You can get to this dialog from the Find File item on the File menu, or by typing in the built-in FIND command

**FILE ASSOCIATIONS:** With File Associations simply type in the filename without the program name, and quite often SmilerShell will know which program to run. For example give "FILENAME.WRI" and SmilerShell knows to run Windows Write on this file. This works very nicely with SHOW or FIND: use SHOW or FIND to pick a file, then press Enter to automatically run the proper program with that file.

**DRAG AND DROP:** SmilerShell supports drag and drop, so you can drop files onto its window from File Manager or any other drag-and-drop server. The filenames will be added at the end of the current command line.

**CURRENT DRIVE/DIRECTORY IN THE TITLE BAR:** It's handy to always know what directory you're in. Toggle this from the Options menu, or just type Alt+D.

**SYSTEM RESOURCES:** Toggle the System Resources display onto the menu bar from the Options menu, or just type Alt+S, to see a real-time running report of your available Windows memory and resources. If you toggle off the menu, the resources report will appear in the command line. Don't worry, nothing you type will be overwritten by the resources report! Just type, your text will reappear exactly as you left it.

**INSERT OR OVERTYPE MODE:** Toggle insert or overtype mode from the Options menu, or just type Alt+O. In overtype mode a flag appears in the title bar.

**TOPMOST WINDOW:** Make SmilerShell a "topmost" window from the Options menu, or just type Alt+T. That way, it's always visible and ready for use, even when you're working in another window.

**GET RID OF THE MENU:** Hit Alt+M to make SmilerShell even smaller. Hit Alt+M again to bring the menu back, or use the Show SmilerShell Menu item on the System menu (the dash thing in the upper left corner).

**SAVE YOUR STACK:** You can save the current command stack to a file and reload it automatically at startup, or at any other time. This gives you a preloaded batch of commands you can search on. The startup loading is set up in the Settings dialog.

## DOS Commands: Fullscreen Or Windowed

The Options menu item **DOS In Window** (Alt+W) controls whether active DOS commands called from SmilerShell run fullscreen or in a window. If you don't specify a preference, then at startup SmilerShell sets the menu toggle to match the current systemwide value set in the "\_default.pif" file in your Windows directory.

To toggle **DOS In Window** for just one command, start the command line with a **right-bracket** (example: >dir \dos). You can use this with the **equals** or **asterisk** flags if you want to (for example =\*>dir).

### **Related Topics:**

[Submitting Commands](#)

[The Options Menu](#)

## The Hide! Menu Item

If you have not disabled the titlebar-button feature, there is a **Hide!** menu item, which hides the command line window. Alt+H works too, handy if the menu has been toggled away.

The exclamation point is a standard Windows flag. It means that there is no actual menu associated with this item, and clicking it will cause immediate action.

### **Related Topics:**

[The Settings Dialog](#)

[Submitting Commands](#)



## SHOW: An Alternative To DIR

The built-in alias **SHOW** is sometimes more useful than DIR. Like DIR, you could issue, say, the command **SHOW \*.WRI** to display all Windows Write files in a list. But unlike DIR, **SHOW** lets you choose one of the files from its list. Click on a filename, then press the OK button, or just double-click on the filename. Either way, the filename you choose is placed on SmilerShell's command line, where it becomes the current in-process command. You can add parameters or edit the command before submitting it.

**SHOW** works nicely with File Associations. Because SmilerShell supports File Associations, quite often you only have to give the filename, without naming the program to run it. SmilerShell can often tell what program to run, just by looking at the filename. So, for example, you could use **SHOW** to select "FILENAME.WRI" into the command line, then just press Enter. SmilerShell knows to run Windows Write on this file.

Maybe you have some other program called **SHOW** that you'd like to run? Since SmilerShell's **SHOW** acts like an alias, you can bypass it by starting the command line with an equals sign.

Another option might be to use SmilerShell's FIND command, which also generates a list of files and lets you pick one.

## Keyboard Shortcuts

These SmilerShell commands work immediately, without going through a menu.

<b>Alt+C</b>	Date/time clock in titlebar
<b>Alt+S</b>	System Resources report in menubar
<b>Alt+D</b>	Current directory in titlebar
<b>Alt+O</b>	Overtyping/insert mode
<b>Alt+T</b>	Topmost window
<b>Alt+L</b>	Show/hide titlebar
<b>Alt+M</b>	Show/hide menubar
<b>Alt+I</b>	Inactive windows stay visible
<b>Alt+W</b>	DOS commands windowed/fullscreen
<b>Alt+F1</b>	Popup Hints

You can display any menu, or use any menu item, from the keyboard. To display a menu press its Alt key combination. When in a menu, press any item's underlined key to run that item.

<b>Alt+F</b>	File menu
<b>Alt+E</b>	Edit menu
<b>Alt+N</b>	Options menu
<b>Alt+A</b>	Favorite Apps menu
<b>Alt+H</b>	Hide the commandline window
<b>Alt+P</b>	Help menu

These standard Windows commands are available in SmilerShell.

<b>F1</b>	Help
<b>ALT+F4</b>	Exit
<b>Alt+Bksp</b>	Undo last action
<b>Ctrl+Z</b>	Undo last action
<b>Shift+Del</b>	Cut selected text, send to the Clipboard
<b>Ctrl+X</b>	Cut selected text, send to the Clipboard
<b>Ctrl+Ins</b>	Copy selected text, send to the Clipboard
<b>Ctrl+C</b>	Copy selected text, send to the Clipboard
<b>Shift+Ins</b>	Paste contents of Clipboard into commandline
<b>Ctrl+V</b>	Paste contents of Clipboard into commandline
<b>Del</b>	Clear selected text
<b>Esc</b>	Clear entire commandline
<b>Ctrl+Esc</b>	Show task list of all running programs (or use the Apps menu, or right-click the titlebar button)

## The Apps Menu

The Apps menu is displayed either when you right-click on the roving titlebar button, or when you select **Apps** from SmilerShell's own menu bar.

The Apps menu is divided into three parts, divided by separator lines. The first part lists four SmilerShell commands. The second part lists all the "favorite applications" you've set up. The third part lists all currently-running tasks. Icons are displayed with menu items when they are available.

The three parts all work the same way. Use your mouse or keyboard to select a line, and that function is executed. It either runs a SmilerShell internal function, or starts the "favorite application," or switches to the chosen currently-running task.

The four SmilerShell commands at the top of the Apps list are **Edit Apps List**, **Find File**, **Command Line**, and **Help**. Just as you'd expect, **Help** displays the SmilerShell Help information. **Command Line** reveals the command line. It's just like left-clicking on the button. but can save you a step in some cases. **Find File** brings up SmilerShell's [Find File dialog](#) so you can search for a file. It also displays the command line if it was previously hidden, making it easier to use the results of your file search.

Finally, **Edit Apps List** lets you modify your "favorite applications" list. You can use **Edit Apps List** to list up to 199 "favorite applications."

To add a new "favorite application," give information for it and click the **Add App** button. Give the text for the Apps menu, and the filename of the program to run. Choose whether, when its menu item is clicked, this command should be copied into the SmilerShell window for editing (perhaps to add parameters), or run immediately. A new item will be added after the currently-selected command, or at the end of the list if no command is currently selected.

You can "drag and drop" a program from File Manager onto the Apps dialog. The name will appear in the "Command to run or fetch" area.

To edit an existing item, select it from the list. The command's information appears in the editing area. Click the **Delete App** button to delete it, or edit the information and press **Change App** to change it.

Programs listed here can use parameters, aliases, commandline flags (\*>=), multiple commands on one line, pipes, redirection, or any other valid options. Anything that works from the SmilerShell command line will work here.

Can't remember a program's filename? Click the **Browse** button to display a dialog box with which you can navigate through your system and find the program name to be used on the command line. In that box, click on a filename to select it into the

**Command to run or fetch line.**

Or press **Find All Apps** to scan all your fixed disk drives and create a list of every program on your system. You can then browse this list and select programs to add onto the Apps menu. Note: unfortunately Windows thinks that a CD drive is the same thing as a hard disk, so if you have a CD, all programs on the current CD disc will be added to the list. Just ignore them (easy to do, since they're sorted by drive).

**Related Topics:**

[The Options Menu](#)

[The FindFile Dialog](#)

## The 'How To Order' Menu Item

Until you order, SmilerShell 's menu bar includes a **How To Order** menu item, which provides information on how to order. Of course, after you purchase SmilerShell, this menu item is superfluous, and goes away.

### **Related Topics:**

[How To Order SmilerShell](#)

## The FindFile Dialog

Can't find a file? Use SmilerShell's built-in FindFile dialog to search by name or contents.

There are two ways to bring up the FindFile dialog. The fastest way is to type the FIND command on the command line, optionally with the target filespec (wildcards are OK) and contents to search for. But to do it this way, you of course must first clear the command line and type the FIND command. What if you don't want to delete what's on the line? In that case you could bring up the FindFile dialog from the File menu. This is useful if your goal is to find a filename and add its name to a command you are in the process of building.

Here are examples of the three ways you might do it from the commandline:

```
find
find *.txt
find *.txt The quick brown fox
```

The first way just brings up the FindFile dialog, with the cursor in its Filename box ready for you to type in a filespec.

In the second way, you have already given a filespec on the command line. It is put into the Filename box for you, and the focus is placed on the Begin Search button. Just hit Enter and off it goes.

The third way is much like the second, but any text after the filespec is placed into the Containing box. Only files that match the filespec **and** contain your text will be shown in the results list. The matching text can be case-sensitive or not, as you choose. Use FindFile's checkbox to indicate your preference.

Of course, after the dialog box appears, you can change the imported commandline text before starting the search.

The target filename can use the usual wildcard characters ? (match one character) and \* (match any number of characters). If nothing is given, \*.\* is assumed. The **Containing** text can be anything you want. There are a number of radio buttons to specify where you want to search. You can restrict the search to the current directory, current directory and all its subdirectories, or all drives whose letters you type in the dialog's drive-list box.

All matching files are listed, with name, size, date, time and attributes. Double-click on a line to change to that directory, or use the buttons to fetch a name into the command line. It will be added at the end of any text already on the line.

**Related Topics:**

FIND: Search For Files On Disk

## **ALIAS and UNALIAS: Create, Change, Remove Aliases**

The ALIAS and UNALIAS commands can create, change, or remove aliases from the command line, one at a time. To work with all your aliases at once, use the [Settings](#) dialog where they are all listed together.

You can use these forms of the ALIAS and UNALIAS commands from the commandline:

**ALIAS** (alone) -- shows the same Aliases dialog available from the [File](#) menu. You can run an alias, or fetch its text into the command line to be edited.

**ALIAS anAlias=itsReplacement** adds an alias to the list, or updates an existing alias.

**ALIAS anAlias** (with no equals sign) displays on the command line the current setting for that alias, ie "anAlias=itsReplacement", allowing you to view it, and perhaps edit and re-submit it. Re-submitting an existing alias with a new replacement will update its entry.

**ALIAS anAlias=** (equals sign but no replacement) will delete that alias from the list.

**UNALIAS anAlias** also deletes that alias from the list.

### **Related Topics:**

[The Settings Dialog](#)

[Aliases](#)



## FIND: Search For Files On Disk

The **FIND** command is one of the two ways to bring up SmilerShell's built-in [FindFile dialog](#). To use it, type the **FIND** command on the command line, optionally with the target filespec (wildcards are OK) and contents to search for. Here are examples of the three ways you might do it from the commandline:

```
find
find *.txt
find *.txt The quick brown fox
```

The first way just brings up the [FindFile dialog](#), with the cursor in its **Filename** box ready for you to type in a filespec. In the second way, you have already given a filespec on the command line. It is put into the **Filename** box for you, and the focus is placed on the **Begin Search** button. Just hit Enter and off it goes. The third way is much like the second, but any text after the filespec is placed into the **Containing** box. Only files that match the filespec **and** contain your text will be shown in the results list. The matching text can be case-sensitive or not, as you choose. Use FindFile's checkbox to indicate your preference.

Of course, after the dialog box appears, you can change the imported commandline text before starting the search.

The target filename can use the usual wildcard characters ? (match one character) and \* (match any number of characters). If nothing is given, \*.\* is assumed. The **Containing** text can be anything you want.

You can also bring up the FindFile dialog with the Find File item on the [File](#) menu.

### **Related Topics:**

[The FindFile Dialog](#)

## HISTORY: Display Command History

Type the command **HISTORY** to display the command-history list. This is the same list that comes up if you click on the Command History Button, or select the **Command History menu item** on the File menu. It shows all commands issued during this session, plus (if you've set this up) all the commands you issued during the last session and any command-stack file you read at startup or from the File menu.

The dialog box that appears has buttons to fetch the text of a command into the edit area, or to submit it to be run again as is. Double-clicking on a command also sends it to be re-run.

## PATH: Change The Search Path

The **PATH** environment variable is a systemwide value listing the directories to search for a program you want to run. Unlike DOS applications, Windows programs cannot usually change the **PATH** environment variable. However, SmilerShell provides you with some measure of ability to do this.

At startup, SmilerShell reads the systemwide search path. You can then use the SmilerShell **PATH** command to change the list of directories listed there. SmilerShell will then search your designated **PATH** for the programs you ask it to run, instead of the systemwide one.

There are three ways to use SmilerShell's **PATH** command:

**PATH** by itself shows the current path in SmilerShell's edit window.

**PATH=something** sets SmilerShell's copy of the path to the "something" you specified.

**PATH=** (with nothing after the equals sign) sets SmilerShell's path to be the same as the current systemwide path.

So, an easy way to edit your path is to do a **PATH** (no parameters) to get the current path into the edit window, then edit it, then just hit Enter to submit the modified path. You can go back to the systemwide path at any time, by doing a **PATH=**.

You can change SmilerShell's **PATH**, but not the **PATH** of any programs called from SmilerShell. These will still have only the original systemwide **PATH**.

### **Related Topics:**

[Submitting Commands](#)

[About Internal DOS Commands](#)

## Built-In SmilerShell Commands

These enhanced commands are built into SmilerShell:

ALARM: Alarm Clock With Reminder Message

ALIAS and UNALIAS: Create, Change, Remove Aliases

DC: Directory Change The Fast Way

FIND: Search For Files On Disk

HISTORY: Display Command History

PATH: Change The Search Path

SHOW: An Alternative To DIR

Using 4DOS/NDOS Internal Commands

## Using 4DOS/NDOS Internal Commands

If you use the 4DOS or NDOS command processor, you can use the **Settings** dialog to set up SmilerShell to correctly handle all the 4DOS/NDOS internal commands that might be useful from a Windows command line. The supported 4DOS commands are shown below.

In some cases, a 4DOS command is the same as a SmilerShell command (alias, history, unalias). For these, start the command with an equals sign to pass it to the command processor unchanged (for example, =history), as if it were a SmilerShell alias. In other cases, the 4DOS command has only marginal use under Windows. Any command, however, that just might possibly be useful has been made legitimate under SmilerShell.

To add a command not on this list, set up a SmilerShell alias that tells the command processor to run your command. For example, if you want SmilerShell to send the command DELAY to the NDOS.COM command processor, set up this alias:

**delay = ndos.com /c delay**

The /c switch tells the command processor to run this one command. Similarly, if you only want to use some of the 4DOS/NDOS commands, turn off SmilerShell's internal support in the Settings dialog, and set up your own "/c" aliases for the commands you want to use.

Supported Internal 4DOS/NDOS Commands: ?, ALIAS, BEEP, CDD, DESCRIBE, DIRS, ESET, FFIND, FREE, GLOBAL, HISTORY, LIST, LOADBTM, LOG, MEMORY, MOVE, POPD, PUSH, REBOOT, SELECT, SETDOS, SWAPPING, TEE, TIMER, TRUENAME, UNALIAS, Y

## The Settings Dialog: Set Preferences

The **Settings** dialog allows you to customize SmilerShell in a number of ways. You can alter its behavior at startup and exit, how it processes commands, its clock formats, aliases, data files, button exceptions, and other options. Let's take it section by section.

**Startup:** By default, at startup SmilerShell puts its titlebar button into the currently-active application and hides its command line. If you'd rather not have a roving titlebar button, un-check the **Show Button** box. This will also eliminate SmilerShell's **Hide!** menu item. After all, if there's no button and you **Hide!** SmilerShell, how would you get it back? For the same reason, if you un-check **Show Button** you must also un-check **Hide SmilerShell**.

Select **Startup DC Scan** if you want SmilerShell to re-scan your **Drives To Scan For DC Data** (see below) whenever you launch it. This is handy if your system's directory configuration changes a lot. It keeps **DC** current with your system.

**Exit:** Check **Confirm** if you want a dialog to pop up to confirm that you really do want to exit SmilerShell. Check **Save State** to have SmilerShell remember from session to session its current directory, current switch settings, the command history for all commands you gave in the current session, and other information.

**Command Processing:** Do you want SmilerShell to use the command processor listed in the COMSPEC environment variable (perhaps 4DOS or NDOS) or explicitly use COMMAND.COM? And do you want SmilerShell to send \$DOS/NDOS internal commands to your chosen command processor?

**Time And Date Formats:** These are used by the clock in SmilerShell's titlebar. Choose your preferred format.

**Miscellaneous:** Three otherwise-unrelated options. **Restore Time** tells SmilerShell how many milliseconds to pause before trying to regain the focus from an "(Inactive..." window. Windows needs a little pause here. How little can your system get away with? Default is one second (1000 milliseconds). **Multi-Command Separator** lets you select the character that divides multiple commands on one line. **Drives To Scan For DC Data** is a list of drive letters, separated by spaces, commas, or nothing, as you prefer. These drives will be scanned, and the names of all their directories saved, whenever you do a DC scan. These will be the directories you can change to with DC.

**Files:** The **DC Data File** is where SmilerShell stores the results of doing a DC scan. By default it is called smishell.dir and is in the same directory as the SmilerShell ini file. If the **Startup Commands File** exists at startup, its commands are preloaded into SmilerShell's command history list, available to be searched for. By default this is called smishell.stk and is in the same directory as the SmilerShell ini file. Both of

these files have a Browse button, allowing you to navigate through your system to find the files you need.

**Button Exceptions (List):** These are apps that you designated with the Button Exceptions menu item. You can change one by simply using the menu item on the same app again, but if you need to delete an exception entirely, you'd do it here. This is just a list, it can't be edited.

**Aliases (Can Be Edited):** These are all your aliases, one per line. You can edit or delete an existing alias, or add new ones. Do you want your aliases to be case-sensitive? Here's where you indicate it.

**Related Topics:**

[The Initialization File](#)

## Choosing Fonts And Colors

SmilerShell lets you choose the command line's font, foreground (text) color and background color. These can be chosen from the Options menu.

**Foreground Color** brings up the standard Windows color dialog. You can set the color of the edit control's text. Pick a standard color, or create a custom color, but note that if you pick a dithered color, Windows uses the nearest solid color instead.

**Background Color** also brings up the standard Windows color dialog, allowing you to set the edit control's background color. Pick a standard color, or create a custom color.

You can get a very interesting effect by choosing a dithered background color. As with **Foreground Color**, if you pick a dithered color, Windows uses the nearest solid color behind actual text characters. But it uses your chosen dithered color in the rest of the edit area. This creates an unusual raised effect as you type. Try it!

**Font** brings up the standard Windows font dialog. You can set the command line font to any available style and size. However, because of Windows limitations, italic fonts don't look as good as bold or normal fonts.



## Using SmilerShell On A Network

SmilerShell will run well in a networked environment. The software can be installed on a server, and every user can have an individual initialization file on their local machines. To set this up, use Program Manager's Properties dialog to give each user a SmilerShell icon that calls the executable program on the server, with a parameter on the Properties command line that uses the /ini= parameter to point to an initialization file on user's own machine. For example, if the server is drive M: the icon's Properties command line might look like this:

<b>Description:</b>	<b>[Fred's SmilerShell Setup</b>
<b>Command Line:</b>	<b>[M:\programs\smishell.exe /ini=c:\smishell.ini</b>
<b>Working Directory:</b>	<b>[c:\smishell</b>
<b>Shortcut Key:</b>	<b>[None</b>

The working directory won't make a lot of difference, since SmilerShell can be configured to start up in the same directory it was in last time it was run.

Of course each network user must have a separate license to use SmilerShell.

Another way to use SmilerShell on a network is to use its **ONECMD** option to perform synchronous network connect/task/disconnect sequences. See [The ONECMD Option](#) for more information on this.

### **Related Topics:**

[The Initialization File](#)

## Command History Button: Lists All Commands

Click on the **Command History Button**, at the right side of the SmilerShell window, to display the command-history list. This is the same list that comes up if you type the command HISTORY, or select the **Command History menu item** on the File menu. It shows all commands issued during this session, plus (if you've set this up) all the commands you issued during the last session and any command-stack file you read at startup or from the File menu.

The dialog box that appears has buttons to fetch the text of a command into the edit area, or to submit it to be run again as is. Double-clicking on a command also sends it to be re-run.

## The ONECMD Option: Set Up Custom Desktop Icons To Do Any Task

Have you ever wanted to set up icons to run collections of commands under Windows? Sort of like a batch file, but in Windows, with any assortment of DOS or Windows commands? And to make things interesting, would you like those commands to run **synchronously** -- the next one doesn't start until the previous one completes?

SmilerShell makes it easy to set up a second (or third, or fourth) SmilerShell icon with a batch of commands that will run whenever you click on it. And don't worry, you can have your regular copy of SmilerShell running at the same time with no side effects.

You might use this on a network to set up **synchronous** connect/command/disconnect tasks, each on their own icon. The **connect** can use a network drive letter, the **command** can be a multi-command SmilerShell alias, and the **disconnect** can free the drive letter again. Because it's SmilerShell, you are guaranteed that the tasks will be done in order -- the next one won't start until the previous one completes. That's just one example of the power of this option.

Here's how to do it.

First, make a second copy of SmilerShell's Program Manager icon. Perhaps the easiest way to do this is to pick up the icon with the mouse (click and hold) while pressing down the Control key, then "drag and drop" this second copy of the icon. You can drop the copy back into the same Program Manager group, or into another group.

Next, highlight the second copy of the icon and bring up Program Manager's **Properties** dialog. It's on the File menu. Use the **/ini=** commandline option to give this new icon a different ini file. Let's call it **c:\onecmd.ini** in this example. You'd set it up like this:

<b>Description:</b>	<b>[OneCommand Sample SmilerShell Setup</b>
<b>Command Line:</b>	<b>[c:\smishell\smishell.exe /ini=c:\onecmd.ini</b>
<b>Working Directory:</b>	<b>[c:\smishell</b>
<b>Shortcut Key:</b>	<b>[None</b>

Exit from any other copy of SmilerShell currently running, and start SmilerShell from the new icon. It will (with your permission) create the new ini file. In this copy of SmilerShell, set up the assortment of commands as a "multiple commands on one line" SmilerShell alias. Let's call the new alias **mybatch**. Then exit from SmilerShell.

Bring up the **Properties** dialog again, and add the **/onecmd=** flag to the commandline, so it looks like this:

**Description:** [OneCommand Sample SmilerShell Setup  
**Command Line:** [c:\smishell\smishell.exe /ini=c:\onecmd.ini  
/onecmd=mybatch  
**Working Directory:** [c:\smishell  
**Shortcut Key:** [None

That's it. Now whenever you launch that icon, SmilerShell will run all the **mybatch** commands, then immediately exit.

You can, by the way, set it up to run any legal SmilerShell command, not just aliases.

## ALARM: Alarm Clock With Reminder Message

SmilerShell's clock includes a companion alarm and reminder system. You can set the alarm for any time in the next 24 hours, and to display any message when it goes off. To set the alarm enter the command **alarm** followed by the time you want it to go off, followed by the message it should display. You can use 12-hour time (12:00 through 11:59 with **am** or **pm**), or 24-hour time (00:00 through 23:59). If you like, you can put an equals-sign between the word **alarm** and the time. Here are some examples of setting the alarm:

```
alarm=3:25 pm Call Bill before he leaves
alarm 11:30 lunch with Peter, Wendy, and the boys
alarm 23:30 ring Fred and Barney in California
alarm = 1:30 pm
```

Remember that 12:00 by itself (without **am** or **pm**) is noon! To specify midnight, use 12:00 am. Or use 0:00 to specify midnight using 24-hour time.

When the alarm is set, an exclamation point will appear in SmilerShell's titlebar. To turn off the alarm, enter the command **alarm=** (an equals sign with nothing after it). To see what the alarm is set to, enter the command **alarm** all by itself.

Since the alarm is part of SmilerShell's titlebar clock, if you turn off the titlebar clock you will turn off any pending alarm too. Toggling off the titlebar itself (and with it the titlebar clock) will also deactivate the alarm. And alarms are not saved from session to session. But minimizing or hiding SmilerShell's commandline will not bother the alarm. As long as SmilerShell (and its clock) is running, the alarm will go off on schedule.

Maybe you have some other program called ALARM that you'd like to run? Since SmilerShell's ALARM acts like an alias, you can bypass it by starting the command line with an equals sign.

