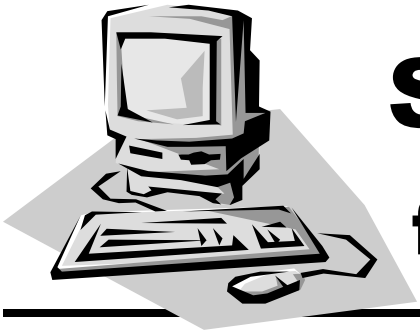


Shorthand for Windows™

A How-to Guide

version 9

by G. C. Lim

3rd Edition

Published by
OfficeSoft LLC, California
Web site: http://www.pcshorthand.com
Email: book@pcshorthand.com

Revision 2005.02.05

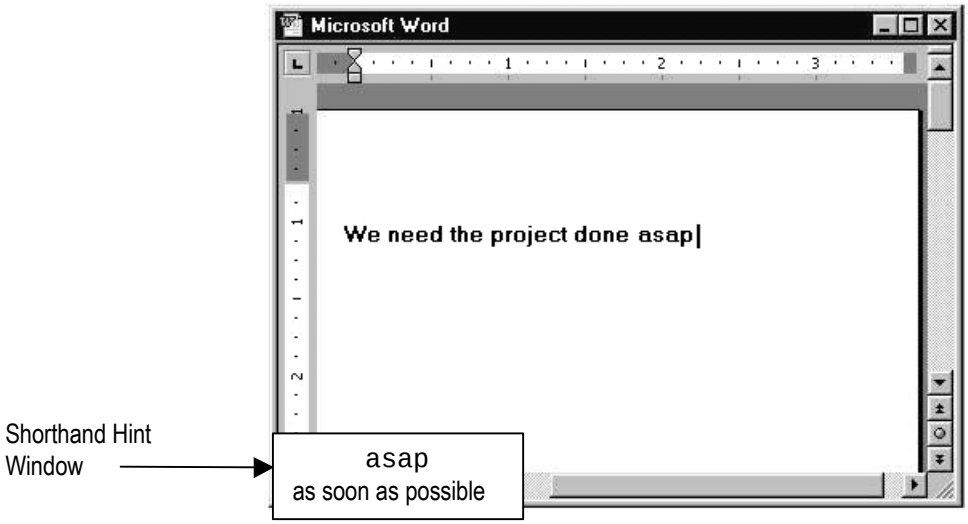
Table of Contents

Chapter 1 Introduction

What is Shorthand?

Shorthand is a word expander and keyboard simulation program that lets you define abbreviations or keywords to play back previously stored words, phrases and even large blocks of text. This reduces keystrokes and increases your typing speed.

Shorthand works with almost all Windows applications that run with the Windows™ desktop operating system (Windows 95/98/Me/2000/XP/Server 2003) and will replace your abbreviations as you type. For example, let's say you defined an abbreviation in Shorthand called `asap` to represent the phrase `as soon as possible`. Whenever you type `asap` , a small, yellow Shorthand *Hint Window* will pop up with the suggested text.



The full text ("as soon as possible") is inserted after the space bar, any punctuation mark or the ENTER key is pressed. You can assign a keyword to represent words, phrases, and sentences. Shorthand also supports dynamic text (i.e. text that can be changed interactively) through Shorthand Tag codes (p. 65). With embedded Shorthand Tag codes, you can tell Shorthand to pause to accept input from the user, insert the current date, synthesize keystrokes, launch other programs or perform mathematical computations.

The number of entries and Shorthand dictionary files is only limited by the size of your hard disk space.

Installing Shorthand

This book assumes you have Shorthand Version 9.00c or later.

Install Shorthand by running SETUP.EXE from the Shorthand CD.

For the latest version of Shorthand, visit

`http://www.pcshorthand.com`

How to use Shorthand

1. Start the Shorthand program (p. 13). If you have a trial license, the About box appears showing the number of days remaining; click OK to continue.
2. Access the Shorthand Main Window by pressing **F10** or clicking on the Shorthand system tray icon (p. 13).
3. Open a dictionary (p. 20). For this example, let's open the DEMO1 dictionary by clicking on the Dictionary tab at the bottom of the Shorthand Main Window labeled "DEMO1" (if there is no DEMO1 tab, choose Open from the File menu and select DEMO1.SPF).
4. Click on the HIDE button (p. 16). The Shorthand Main Window disappears and Shorthand monitors your keystrokes as you type.
5. Run your word processor and open a new document.
6. In your word processor start typing normally; Shorthand will replace abbreviations as you type. For example, if you type:

   `asap` SPACE

   Shorthand should insert `as soon as possible` in your document. (If nothing happens, see p. 129 for troubleshooting tips.)

Tutorials

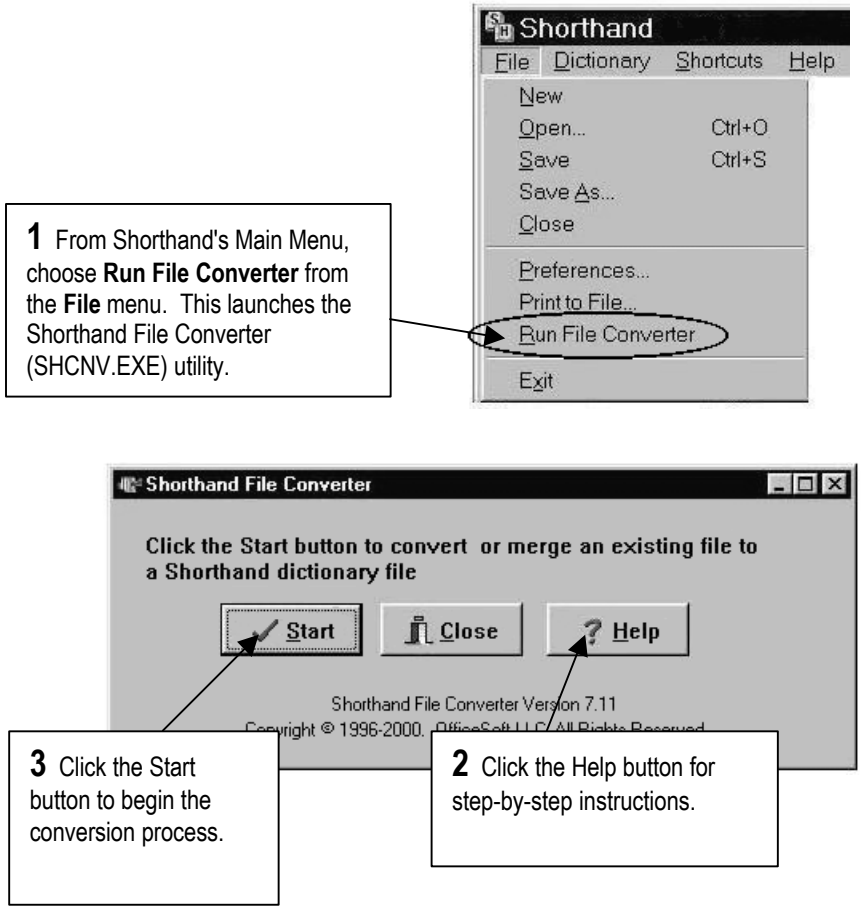
To get a better idea of how Shorthand works, we suggest you go through the tutorials that come with the Shorthand program. To access the tutorials, choose **Contents** from Shorthand's **Help** menu. From the **Contents** tab, choose **Tutorial Demos** then **Quick Tutorial.**

Always keep a backup of your important files separate from your computer.

It is easy to accidentally delete files by pressing on the wrong button or selecting the wrong command. For this reason, you should always keep a copy of your Shorthand dictionary files and other important files on a diskette or CD. Your Shorthand dictionaries are kept in files ending with .SPF and you can use the Windows Explorer (p. 11) program to copy the files to a diskette.

File Converter Utility (SHCNV.EXE)

Shorthand comes with a file converter utility (SHCNV.EXE). To run this utility:



As of this writing the SHCNV.EXE utility can:

- Convert text files into Shorthand dictionary (.SPF) files.
- Convert Microsoft® Word™ (Versions 6.0 and later) AutoCorrect entries to Shorthand format.
- Convert other word expander formats such as Productivity Plus™ (PRD+), Smartype™ and Abbreviate™ files into Shorthand format.
- Merge two Shorthand dictionaries

In general, the procedure is to first save your word list as a text file (using your expander's Print to File or Export to File function) then use SHCNV to convert the text file to a Shorthand .SPF file. Click on the Help button (step 2 above) for specific instructions. If your file format is not supported by SHCNV, you can write a Tcl script to do the job (see p. 118 for an example).

Terminology

Click and Drag

In Windows, you can move or resize some objects by *clicking and dragging*. When you are asked to "click and drag" an object, you first select the object by pointing your mouse arrow on the object and clicking your left mouse button on it. With the mouse arrow still on the object, hold down the left mouse button and move your mouse to *drag* the object around the screen.

Dictionary

Shorthand stores your list of abbreviations and text in a *dictionary*. On your hard disk the dictionary is saved in a file ending with **.SPF**. You can create as many Shorthand dictionaries as can fit on your hard disk but only one dictionary can be active at any time (but you can *link* dictionaries together (p. 40) and let Shorthand access the linked dictionaries as if they were one big dictionary file). The number and size of Shorthand dictionaries are limited only by the hard disk space available on your computer. Other word expanders may use the terms "word list," "glossary," or "abbreviation list" to what Shorthand refers to as a dictionary.

Dictionary Tabs

Dictionary tabs (p. 17) are located at the bottom of the Shorthand Main window and allow you to quickly open a dictionary. You create a dictionary tab by creating (p. 18) or opening (p. 20) a Shorthand dictionary file. You remove a tab by closing the file (p. 45).

Double-clicking

Double clicking on an object means clicking your left mouse twice (in rapid succession) on an object.

File Shortcut

A *File Shortcut* (p. 42) is a keyboard command (e.g. **Ctrl+F1**) you assign to open a Shorthand dictionary, launch an external file or run a Tcl script.

Hint Window

When enabled, Shorthand displays a small yellow *Hint Window* (p. 58) whenever you type a recognized short form. The Hint Window usually appears at the bottom of the text window in which you are currently typing.

Hot Key

The Hot Key is a special keystroke you type from within your word processor to call up Shorthand. By default, the Hot Key is the **F10** function key. To change the Hot Key, see p. 49.

Icon

An *icon* is a small picture that represents a file, program or command. When you start Shorthand, Shorthand places a small icon of itself in the Windows *System Tray* on the lower right hand corner of your screen (p. 13). You can bring up Shorthand by clicking on its icon.

Keyboard command / Keyboard shortcut

Keyboard commands are commands issued through a keystroke combination.

For example, **Alt+Tab** (while holding down the ALT key, press the Tab key) will switch Windows applications and **Alt+F4** will close the current window.

Keyboard shortcuts refer to the keystroke combination that you can use instead of using the mouse to click on the menus. For example, in Microsoft Word, the keyboard shortcut to turn bold on/off is **Ctrl+B**. Since most functions of your word processor can be assigned a keyboard shortcut, Shorthand can access your word processor's special functions by simulating the appropriate keystroke combination with Shorthand's @KEY tag (p. 68).

For a list on Windows keyboard commands and keyboard shortcuts see p. 132. To find the keyboard shortcuts for your word processor, consult your software's documentation.

Keyword

This is the abbreviation or short form (p. 27) you create to represent a longer string of text or keyboard commands.

Keyword Shortcut

A *keyword shortcut* (p. 28) is a keyboard combination you can assign to a keyword. For example, if you assign the **F9** key as the keyword shortcut for the keyword `asap` to represent the phrase "as soon as possible", pressing **F9** will insert "as soon as possible" into your word processor.

Links

When you link files (p. 40), you are connecting them together with the active dictionary. Shorthand allows one active dictionary at a time, but the linking feature lets you have access to more than one dictionary file by connecting them to the active dictionary.

Menus

Menus are lists of commands. The *Main Menu* is located at the top of the Shorthand window with the command categories (File, Dictionary, Shortcuts, Help) listed on it. A *Pop-Up Menu* appears when you click your right mouse button inside a window; for example, clicking your right button in a text window can show you a menu to cut, copy or paste selections.

Shorthand Folder

The folder on your hard disk where the Shorthand program files were installed. For Shorthand Version 9 this folder is normally

```
C:\Program File\Shorthand for Windows\Sh9
```

If you don't know the name of the folder, you can use Windows Explorer to search for the file **sh9.exe** which is the name of the Shorthand program file.

Shorthand License File (SH9.LIC)

When you purchased a Shorthand license, you were given a *Registration ID* (p. 47) which, when entered into Shorthand, converts Shorthand into a licensed version. After you enter the Registration ID, Shorthand saves your registration into a file called SH9.LIC in the Shorthand folder. To remove a license from the computer, simply delete the SH9.LIC file; you should remove your license from a computer you no longer plan to use to prevent others from illegally using your Shorthand license.

START button and Taskbar

At the bottom of your Windows desktop is the *Taskbar* which displays icons of active programs. Shorthand normally does not display its icon in the Taskbar except when one of its windows (e.g. About box) is visible. If the Taskbar is not visible, moving your mouse to the bottom of the screen should make it appear. It is at left side of the Taskbar where you can find the Windows **START** button. With the **START** button, you can start a program or find a file quickly (p. 13).

System Tray

The *System Tray* is found at the right side of the *Taskbar* (on the lower right corner of your screen). The Shorthand icon can be accessed from here (p. 13).

Suggestion Window Shortcut

When the *Suggestion Window* is enabled (p. 56), it displays up to nine choices that is closest to the word being typed. The function keys (F1 to F9) are used as the *Suggestion Shortcut* for selecting the choices in the *Suggestion Window*.

Tags

Shorthand *Tags* (p. 65) refer to special commands that you can embed in your Shorthand long forms to do such things as type out the current date, synthesize a keystroke, display a pick list, import a text file or run a Tcl script.

Tcl

Tcl stands for "Tool Command Language" and is the scripting language used by Shorthand (see p. 89). Tcl is a popular language with powerful commands to do such things as arithmetic computations, string manipulation and regular expression parsing. Teaching you the details of Tcl programming is beyond the scope of this book but you can find out more about Tcl from the Internet at http://www.scriptics.com.

Text to Type

The *Text to Type* (also called "long form" or "macro") refers to the text and keyboard commands that will be typed out and inserted into your document (see p. 30).

Title Bar

The *Title Bar* is the top line of your application's window that contains the name of the software program. Shorthand's Title Bar also shows the name of the active dictionary file.

Variable

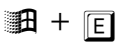
When creating Shorthand Tags, you have the option of defining a variable (p. 87) that will hold the results of the Tag. For example, if you assign a variable called `{colors}` to a pick list of colors**,** Shorthand will replace all instances of `{colors}` with the items chosen by the user from the pick list.

Windows Desktop

The large empty area you see when you first start Windows is called the *Windows Desktop*. When you install Shorthand, you have the option of creating a Shorthand shortcut icon on the desktop; double clicking on this icon will launch Shorthand (p. 13).

Windows Explorer

Windows Explorer is a file management tool included with Windows which allows you to copy, delete and rename files and folders on your hard disk. You launch Windows Explorer by pressing



(i.e. while holding down the Windows key, press the **E** key).

See your Windows user manual for instructions on how to use Windows Explorer.

How to use this book

Menu commands

Menu commands will be indicated by **bold** text with an ⇨ symbol. For example, **Dictionary ⇨Show Shortcuts** means choose **Show Shortcuts** from the **Dictionary** menu.

Keyboard combinations

The command for keyboard combinations will appear with **Ctrl**, **Shift** and/or **Alt**, the + sign and the letter command. For example, when you see:

Ctrl+G

This means that you should go to your keyboard and, while holding down the CTRL key, press the G key. These two keys should be pressed simultaneously to be considered a keyboard combination. Do not press the + key.

Text to Type box

You will need to be especially careful when entering text in the Text to Type box as every character, space and line in the Text to Type box will be processed by Shorthand. In this book, words to be typed in the Text to Type box will look like this:

Text to Type:

```
Type this in the Text to Type box ENTER
Line 2 ENTER
Line 3 ENTER
Line 4: This is a very long line that cannot fit on one line and
continues to the next line.ENTER
Line 5
```

The ENTER symbol represents a "hard return" created by pressing the **Enter** or **Ctrl+Enter** key on your keyboard. Note that Line 4 in the example above is one long line that word wraps to the next screen line. Also note that Line 5 does not end with a hard return (do not press ENTER at the end of Line 5). A common error is to put an extraneous hard return at the end of the Text to Type which will result in Shorthand typing out an extra line in your word processor.

Tips and Notes

Important information are marked with a ☞.

Boxes with a ⓘ contain helpful suggestions and tips.

Chapter 2 Basic Operations

How to start Shorthand

To start Shorthand, click on Windows **START** button:

then choose **Programs ⇨ Shorthand 9 ⇨ Shorthand 9.x Application**.

If you want Shorthand to automatically start up whenever you launch Windows, place the Shorthand Program icon in the Windows **StartUp** folder (see your Windows documentation on how to add a program to the StartUp folder).

How to bring up Shorthand's Main Window

All the Shorthand functions, properties and word lists are accessible through Shorthand's Main Window. After starting Shorthand, you can bring up the Main Window as follows:

Method 1: Click on the Shorthand System Tray Icon.

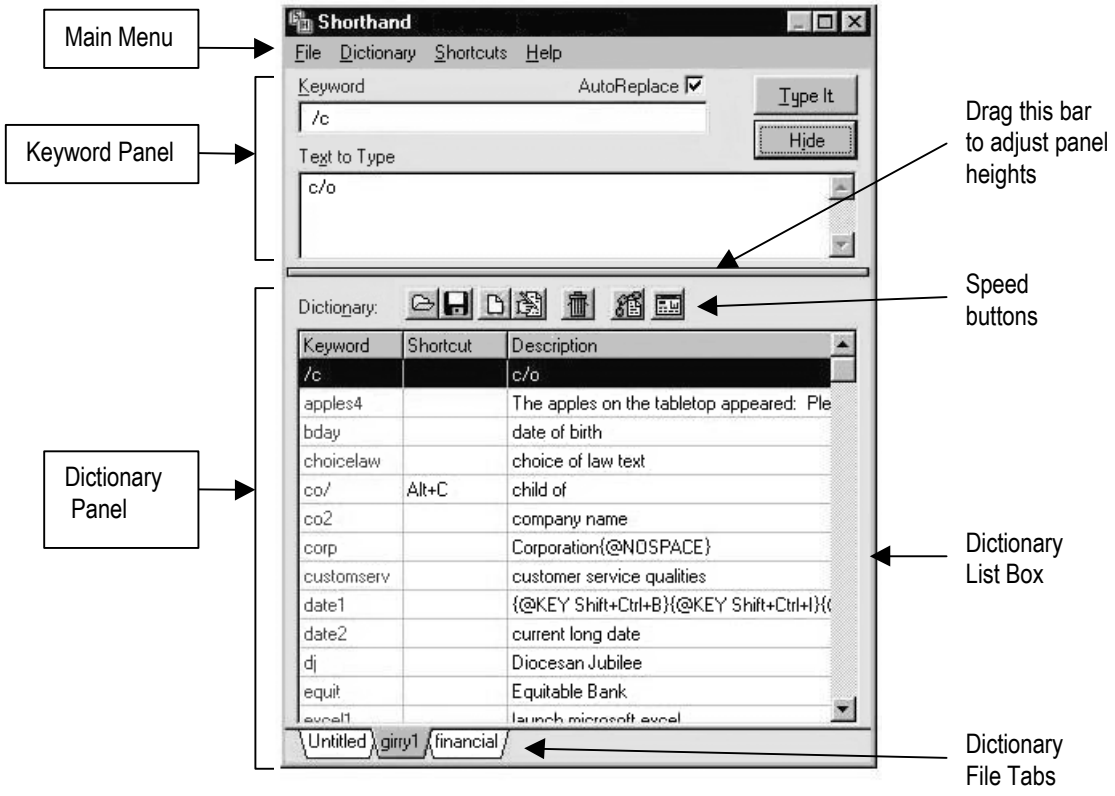
After starting Shorthand, you can access the Shorthand's Main Window by clicking on the Shorthand icon in the *System Tray* on the lower right corner of your screen:

Method 2: Press the Shorthand Hot Key 

Another quick way to access Shorthand's Main Window is by pressing the Shorthand *Hot Key* from your keyboard. By default, Shorthand assigns **F10** as the *Hot Key* (see p. 49 if you wish to change this).

The Shorthand Main Window

Shorthand's Main Window can be divided into three sections: Main Menu, Keyword Panel and Dictionary Panel.



The Main Menu

The *Main Menu* contains these four top-level menus:

File menu

This menu contains commands to create, open, save and print dictionary files. Shorthand dictionary files are saved on your hard disk with an **.SPF** extension. This menu also contains commands to access Shorthand preferences and run the file converter.

Dictionary menu

This menu contains commands to create, modify or remove entries in the active dictionary file. In this menu, you can also search for specific entries in dictionary files as well as copy entries.

Shortcuts menu

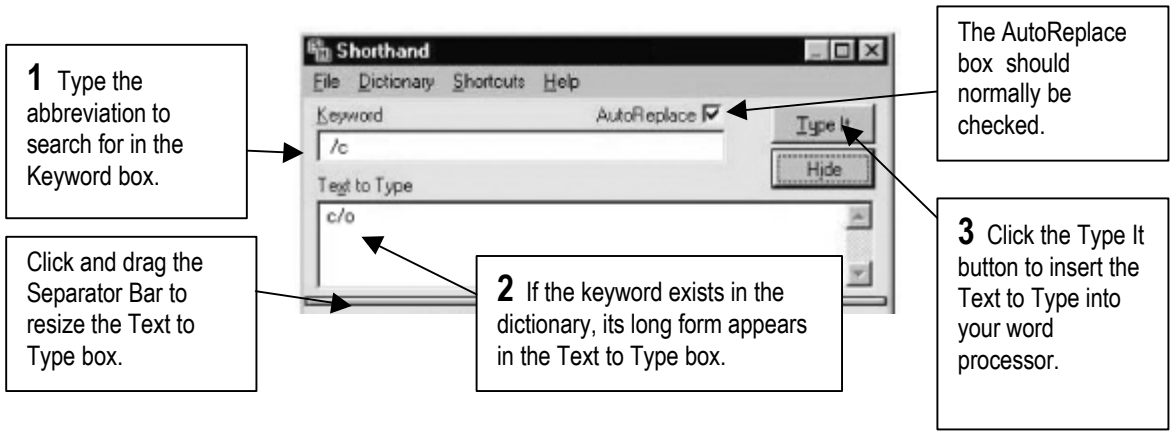
This menu displays your Shorthand *File Shortcuts*. A File Shortcut is a keystroke combination to open a Shorthand dictionary, launch another application or run a Shorthand Tcl script (see p. 42).

Help menu

This menu provides on-line documentation on Shorthand and usage statistics.

The Keyword Panel

Located right below the Main Menu, the *Keyword Panel* contains boxes for the Keyword and Text to Type.



Keyword box

To look for a keyword, type it into the *Keyword* box and the closest matching entry appears in the Dictionary list box. You can press the Up and Down arrow keys in the Keyword box to scroll to the next keyword in the list.

AutoReplace checkbox

If this box is checked, *AutoReplace* will be enabled and Shorthand will automatically detect and replace keywords as you type in your word processor. Since this is what you normally want Shorthand to do, always make sure this box is checked.

Text to Type box

When you type a recognized keyword in the Keyword box or select an entry from the Dictionary list box, the text associated with the keyword appears in the *Text to Type* box. You can edit or enter additional text in the Text to Type box before clicking on the Type It button. Changes you make in the Text to Type box are temporary and will not be saved. To change the size of the Text to Type box, click and drag the *Separator Bar* below the Text to Type box.

TYPE IT button

Clicking this button will command Shorthand to insert the contents of the Text to Type box into your word processor. The TYPE IT button is enabled only if Shorthand was activated from your word processor with the Hot Key (p. 49); the reason for this is by design: Shorthand needs to know where to insert the text and pressing the Hot Key (**F10** by default) tells Shorthand exactly where to place the text. If you had activated Shorthand by clicking on its icon in the System Tray, the TYPE IT button is disabled because Shorthand doesn't know which window to insert the text.

☞ Pressing the ENTER key in Shorthand's Main Window is equivalent to clicking on the Type It button.

HIDE button

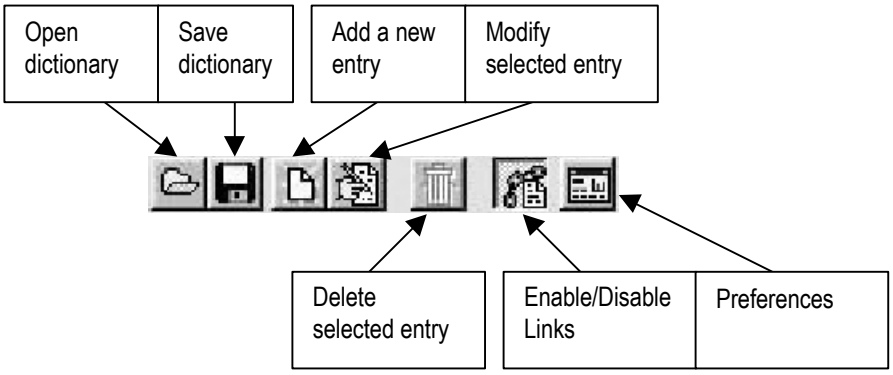
When you click on this button, the Shorthand Main Window will minimize to the *System Tray* (p. 10). If the AutoReplace box next to the Keyword box is checked, Shorthand will replace abbreviations as you type.

☞ Pressing the ESC key in Shorthand's Main Window is equivalent to clicking on the HIDE button.

The Dictionary Panel

Located right below the Keyword Panel, the *Dictionary Panel* is where you manage your word lists.

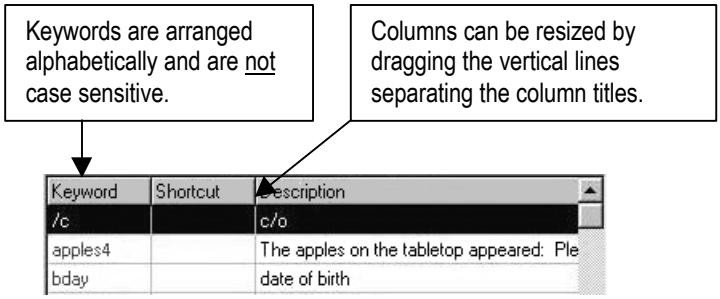
Speed buttons



With a single click, the speed buttons conveniently let you access common Shorthand operations.

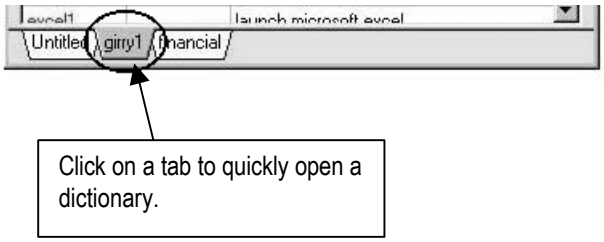
☞ To determine the function of a button, move your mouse pointer over a speed button.

Dictionary list box



The Dictionary list box displays the keywords, shortcuts and their associated text in the active and linked dictionaries. The *Shortcut* column can be enabled/disabled by choosing **Dictionary ⇨Show Shortcuts**.

Dictionary File Tabs



The *Dictionary file tabs* below the Dictionary list box display previously opened dictionary files; you can think of the tabs as a "file history" list. The tabs give you quick way to reopen a dictionary and make it the active dictionary.

To create a new tab:

Open the dictionary by choosing **File⇨Open** or **File⇨New.**

To remove a tab:

Click on the tab to make it active then choose **File⇨Close**.

To reorder tabs:

Click and hold down the left mouse button on the tab and drag the tab to the new position.

☞ While you can have many dictionary files displayed on the tabs, Shorthand only allows one active dictionary at a time. To access the contents of two or more dictionaries simultaneously, you will have to *link* them to the active dictionary (see p. 40).

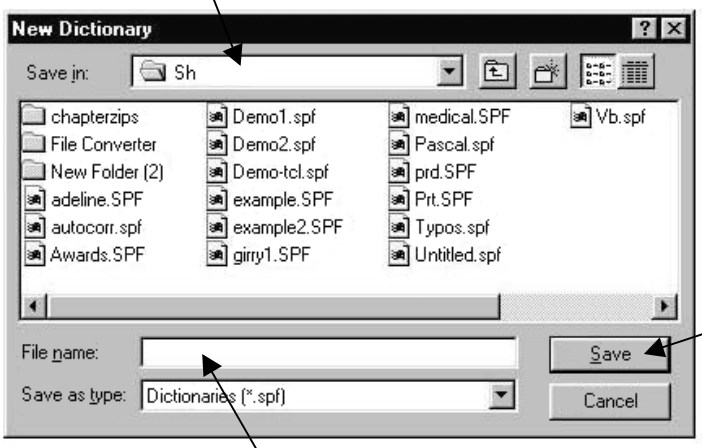
How to create a dictionary

Shorthand stores your word lists in *dictionaries*. Each dictionary is saved to a standard Windows file with the extension **.spf**; you can create as many dictionaries as can fit on your hard disk and moving your dictionaries to other computers is as simple as copying a file.

To create a dictionary in Shorthand:

1 From Shorthand's Main Menu, choose **File⇨New.** The New Dictionary window appears.

2 By default, Shorthand saves your dictionaries to the Shorthand folder. To save to a different folder, click in the **Save In** box and select a new folder.



4 Click **Save** to create the file. Shorthand's Main Window reappears and you should see a new dictionary tab with the dictionary name.

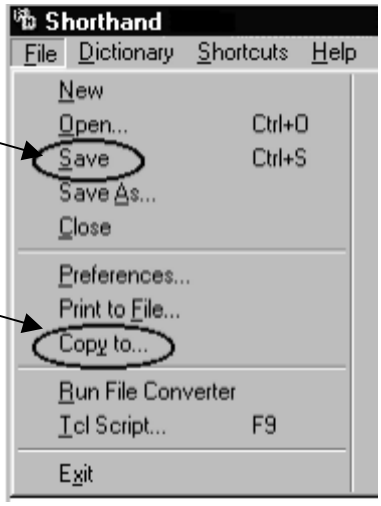
3 Type the new dictionary name in the **File name** box.
Warning: If you specify an existing filename, Shorthand displays a warning message and asks if you really want to replace the existing file. If you answer **Yes** all entries in the file will be erased and will be lost (but may be recovered if Shorthand had created a backup; see p. 19).

How to save a dictionary

When you make changes (i.e. add new entries, modify an entry, etc.), your modifications are not made permanent until the dictionary is *saved* to the hard disk. You can command Shorthand to save your dictionary as follows:

To save changes and update your dictionary file, choose **File⇨Save.**

To make a duplicate copy of the dictionary to another folder or disk, choose **File⇨Copy To.**



Automatic saving

By default, Shorthand automatically saves your dictionary files when you click on the **Hide** button, open another dictionary or shut down Shorthand. You can enable or disable the automatic saving through the Preference window (p. 62).

Backup files

When a dictionary is saved, Shorthand automatically makes a backup copy of the original file. The backup copy is saved to a file with extension ".~SP". You can restore the original data by using Windows Explorer (p. 11) to rename the backup file to its original name.

Backing up to a diskette

You can make a copy of the dictionary to a floppy diskette or another folder by choosing **File ⇨Copy To**.

Always keep a backup of your important files separate from your computer.

It is easy to accidentally delete files by pressing on the wrong button. For this reason, you should always keep a copy of your Shorthand dictionary files and other important files on a diskette or CD. You can use the Windows Explorer (p. 11) program to copy files to a diskette. Your Shorthand dictionaries are kept in files ending with .SPF.

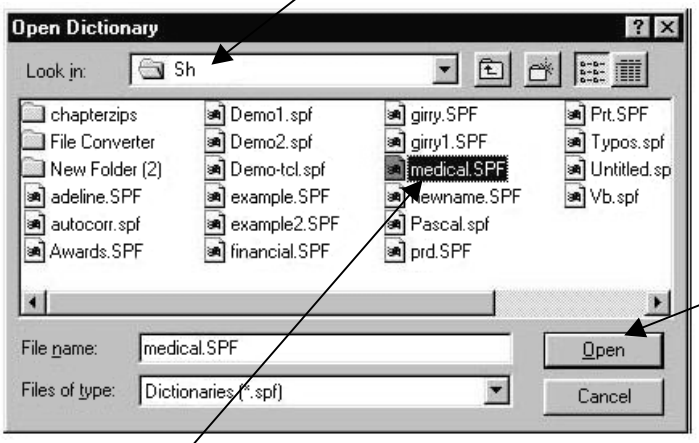
How to open an existing dictionary

Shorthand can only recognize abbreviations that are in the active dictionary. To make a dictionary active, you have to *open* it.

To open a dictionary:

1 From Shorthand's Main Menu choose **File⇨Open**. The Open Dictionary dialog box appears.

2 By default, Shorthand looks for dictionaries in the Shorthand folder. To look in a different folder, click in the **Look In** box and select a new folder.



3 Select the dictionary to open. Shorthand dictionary names end with the extension ".SPF".

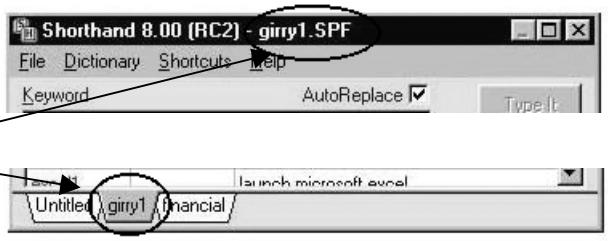
4 Click **Open** to load the selected file. Shorthand's Main Window reappears and you should see a new dictionary tab with the dictionary's name.

☞ Dictionary tabs (p. 17) allow you to quickly reopen Shorthand dictionaries. When a dictionary is created or opened, a tab will appear in the Shorthand Main Window. To reopen a dictionary, simply click on its tab.

How to print a dictionary

To print the Shorthand dictionary word lists, you have to first convert the dictionary to a text file and use your word processor to print the text file:

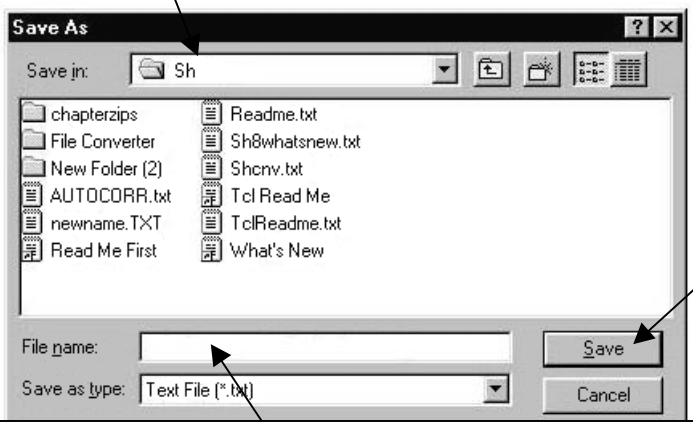
1 Open the dictionary you want to rename (p. 20). The dictionary's name should appear in Shorthand's title bar and be the selected dictionary tab.



2 From Shorthand's Main Menu, choose **File⇨Print to File**. The Save As dialog box appears.

See p. 115 for a Tcl script that can export your word list to a comma-delimited file.

3 By default, Shorthand saves your dictionaries to the Shorthand folder. To save to a different folder, click in the **Save In** box and select a new folder.



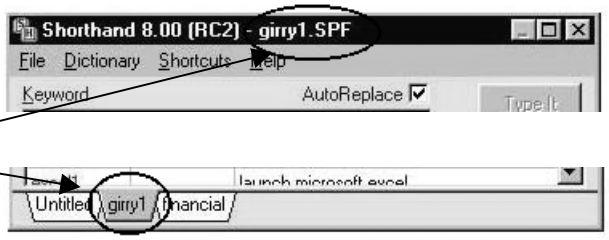
5 Click **Save** to print the dictionary to the text file. You can then open the text file in your favorite word processor program (e.g. Microsoft Word) and print out the word list from there.

4 Type the name of the destination text file in the **File name** box. ***Warning:*** If you specify an existing filename, Shorthand displays a warning message and asks if you really want to replace the existing file. If you answer **Yes** all entries in the selected file will be erased and will be lost.

How to rename a dictionary

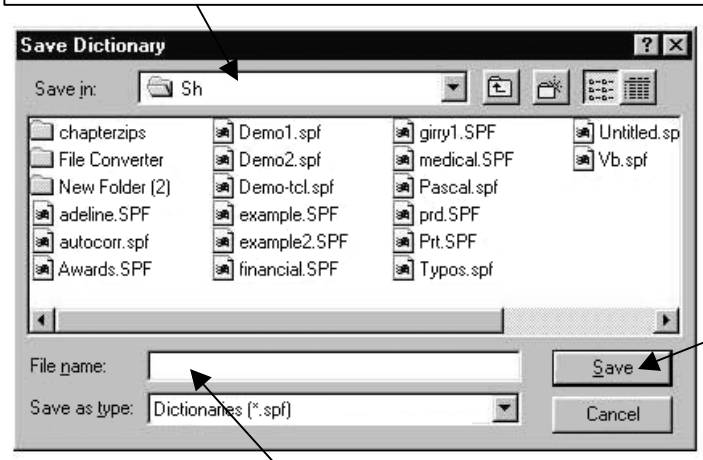
You can change the name of an existing dictionary in Shorthand by saving it under a different name. To save a Shorthand dictionary under a different name:

1 Open the dictionary you want to rename (p. 20). The dictionary's name should appear in Shorthand's title bar and be the selected dictionary tab.



2 From Shorthand's Main Menu, choose **File⇨Save As.** The Save Dictionary dialog box appears.

3 By default, Shorthand saves your dictionaries to the Shorthand folder. To save to a different folder, click in the **Save In** box and select a new folder.



5 Click **Save** to create the new file. Shorthand's Main Window reappears and you should see a new dictionary tab with the dictionary name.

4 Type the new dictionary name in the **File name** box..
Warning: If you specify an existing filename, Shorthand displays a warning message and asks if you really want to replace the existing file. If you answer **Yes** all entries in the file will be erased and will be lost (but may be recoverable if Shorthand had created a backup; see p.19).

Notes

- When you use the Save As command, Shorthand creates a duplicate file; the file with the old name will remain on your hard disk. You can use Windows Explorer (p. 11) to remove unneeded files.
- Since dictionaries are stored as normal Windows files, you can also use Windows Explorer to rename the file; if you do this, you will need to choose **File⇨Open** from Shorthand's Main Menu to reload the file (p. 20).

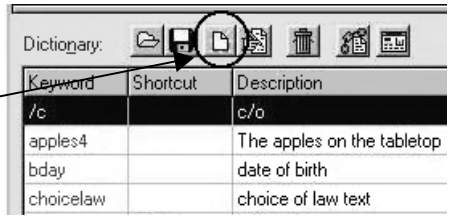
How to add new entries to a dictionary

Use any one of the five methods below to create a new entry in the active dictionary.

Method 1
From Shorthand's Main Menu, choose **Dictionary⇨Add**.

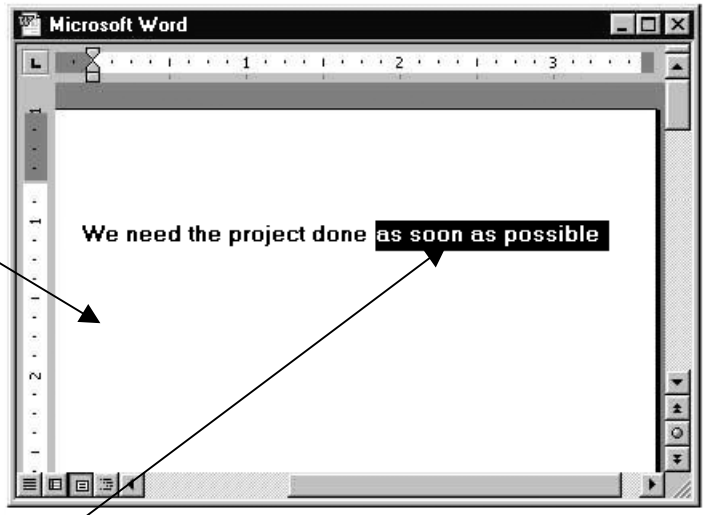
Method 2
Press the INS key on your keyboard anywhere in Shorthand's Main Window.

Method 3
Click on the third speed button located above the dictionary list box.



Method 4
In your word processor, press **Alt+Ins**.

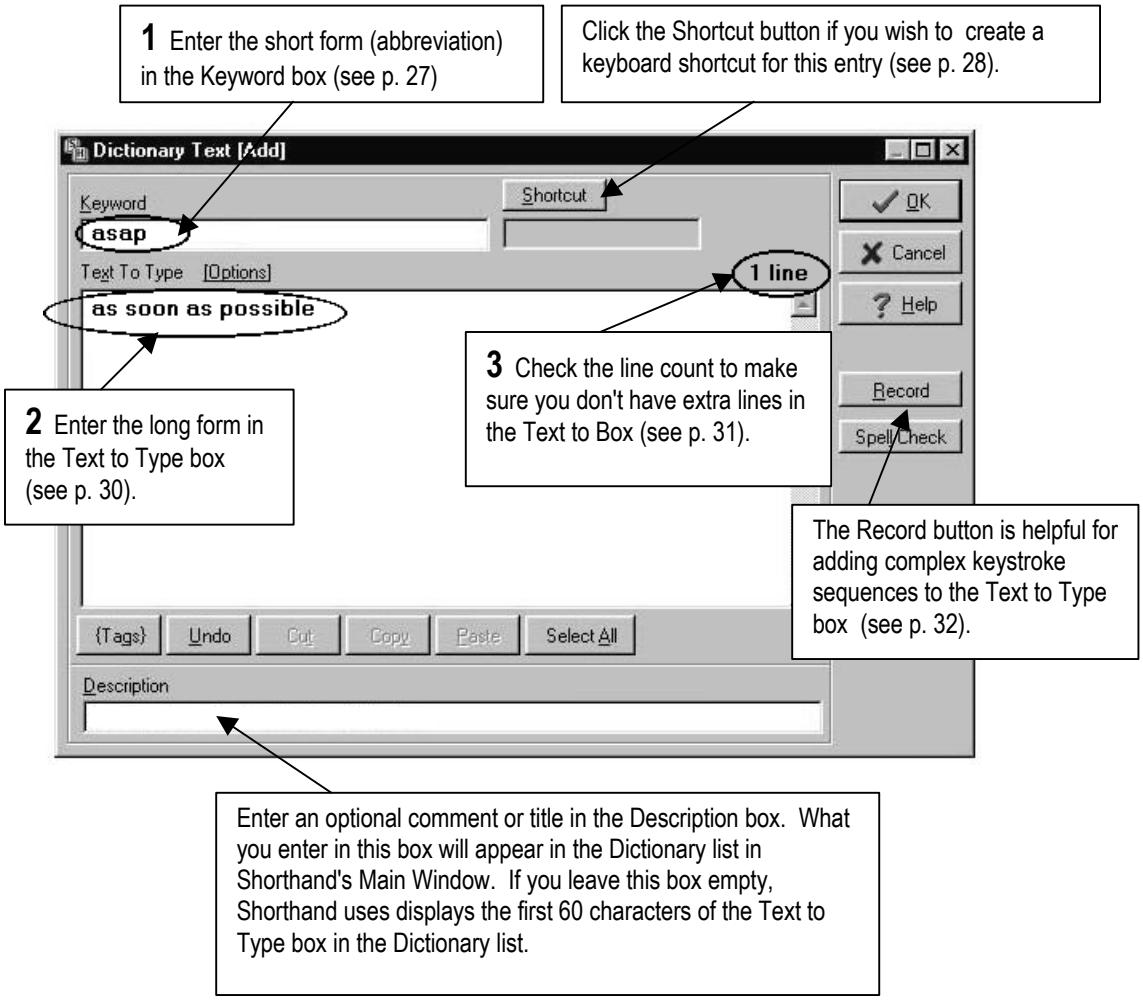
NOTE: For this to work, **AutoReplace** and the **Alt+Ins** options must be both enabled (see p. 55).



Method 5
In your word processor, highlight the block of text you want to add then press **Ctrl+Ins** twice (i.e., while holding down the CTRL key, press and release INS then press and release INS a second time).

NOTE: For this to work, **AutoReplace** must be enabled (p. 51). This feature also requires that your word processor recognizes the **Ctrl+Ins** command to copy text to the clipboard.

When you create a new entry, Shorthand displays the *Dictionary Text [Add]* dialog box for you to enter the Keyword (short form) and Text To Type (long form).



☞ See *Appendix C: Tips for Touch Typists* (p. 131) for an example of how to efficiently add entries while you type.

How to delete dictionary entries

You can remove (delete) an entry from your dictionary as follows:

1 Bring up the Shorthand Main Window (p. 13) and select the entry you wish to delete from the Dictionary list box.

2 Use any one of the three methods below to create a new copy of the selected entry.

Method 1

From Shorthand's Main Menu, choose **Dictionary⇨Remove (Cut)**.

Method 2

Press the DEL key on your keyboard.

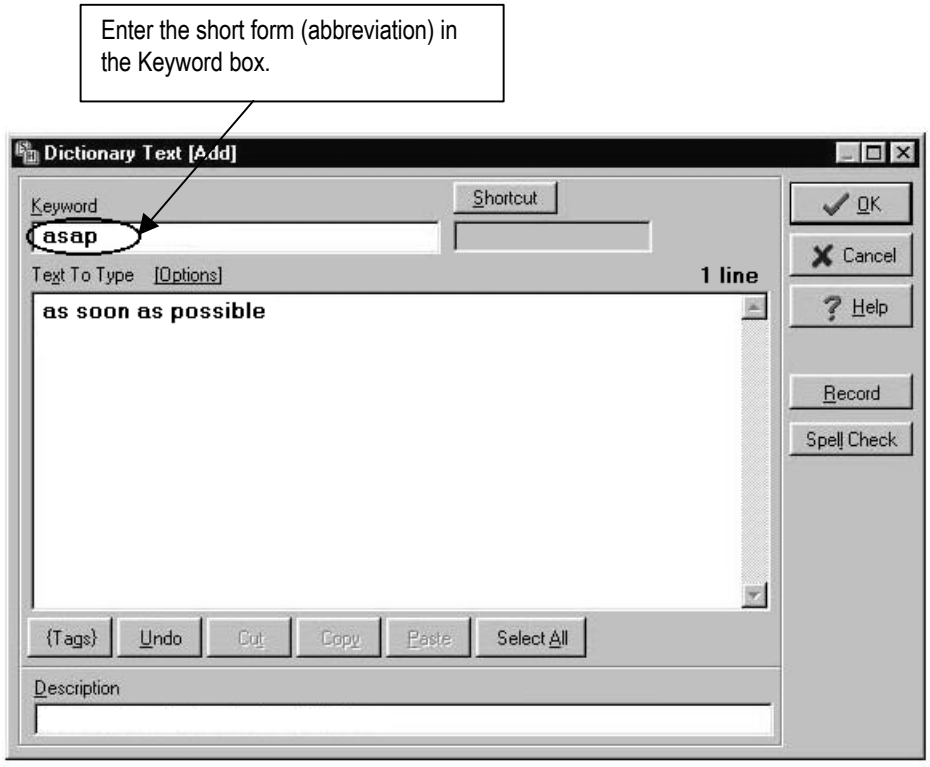
Method 3

Click on the "trash can" speed button above the Dictionary list box.

☞ The deleted entry is copied to the clipboard so, if you make a mistake, you can undelete an entry by choosing **Dictionary⇨Paste**.

How to define short forms or abbreviations

When you add a new entry (p. 24), Shorthand displays the *Dictionary Text [Add]* window. Enter the short form (abbreviation) in the **Keyword** box.



Notes

- Keywords are limited to 32 characters and can contain any printable character and spaces.
- Keywords are not case sensitive.
- If you want the keyword to be replaced as you type, the keyword should not contain spaces. Or, to put it in another way, *keywords with spaces will never be automatically expanded as you type* (but can still be manually expanded by selecting the keyword from the Suggestion Window (p. 56) or from the Dictionary list and clicking on the Type It button).
- Keywords must be unique; a dictionary cannot have two entries with the same keyword.

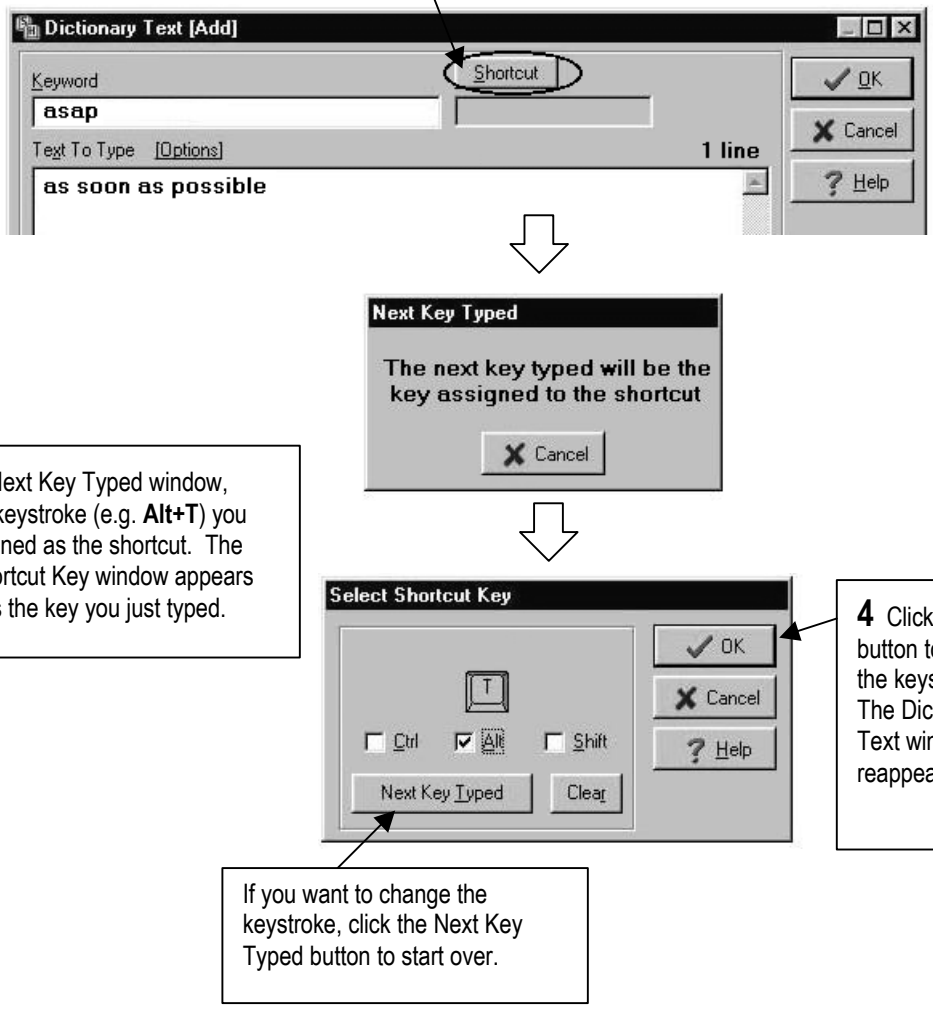
How to define a keyword shortcut

You can further decrease keystrokes by assigning a single keystroke to expand the keyword. Whenever you press a recognized keyword shortcut, Shorthand will immediately insert the keyword's Text to Type into the your word processor.

To define a keyword shortcut:

1 Bring up the Dictionary Text window (see p. 24 if you are creating a new entry or p. 34 if modifying an existing entry).

2 Click the Shortcut button. The Next Key Typed window appears.



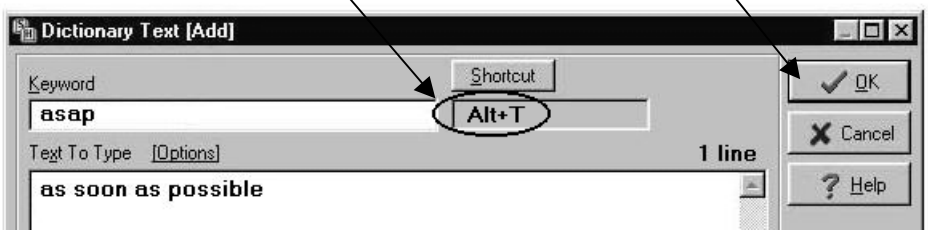
3 In the Next Key Typed window, press the keystroke (e.g. **Alt+T**) you want assigned as the shortcut. The Select Shortcut Key window appears and shows the key you just typed.

4 Click the **OK** button to accept the keystroke. The Dictionary Text window reappears.

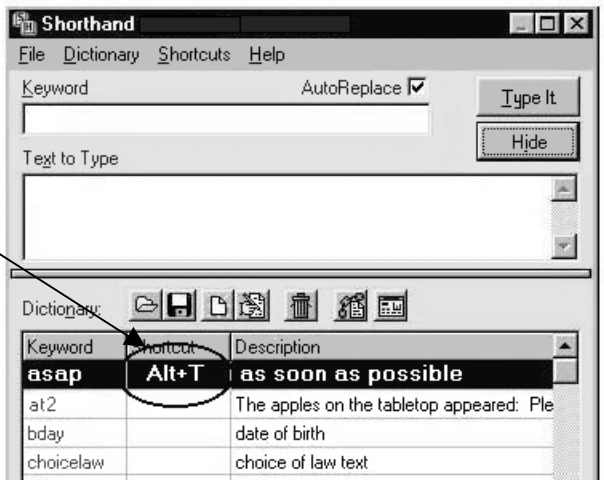
If you want to change the keystroke, click the Next Key Typed button to start over.

The shortcut you chose will appear in the Dictionary Text window under the Shortcut button.

5 Click OK to save the entry.



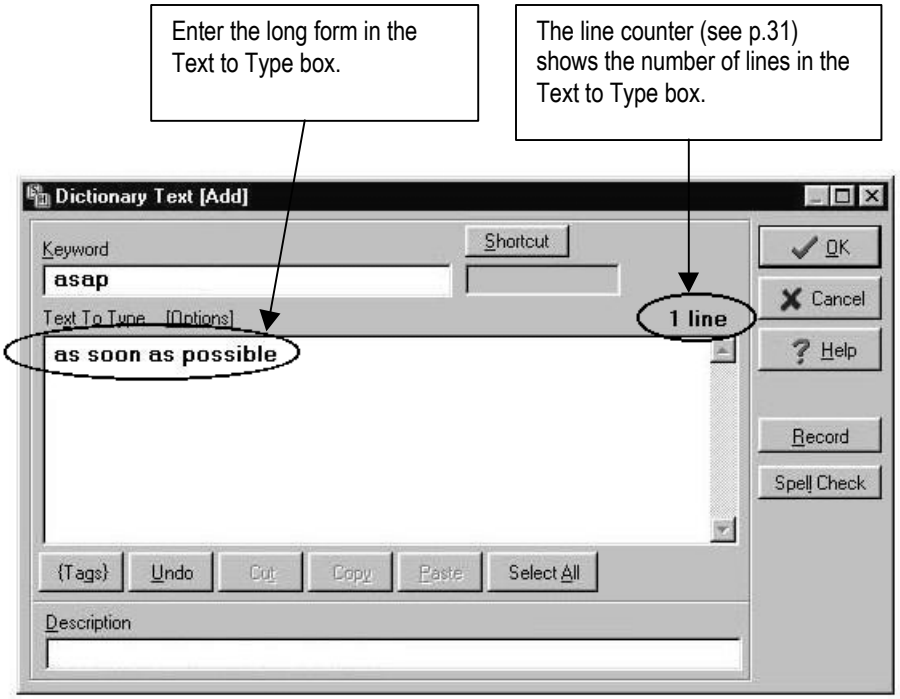
The Shortcut appears in the Dictionary list box. Whenever you press the Shortcut key (**Alt+T** in this example), Shorthand will insert the long form into your word processor.



If you want to use function keys (**F1**, **F2**, etc.) as keyword shortcut keys it is recommended that you combine them with **Ctrl**, **Shift**, and/or **Alt** to avoid conflicts with the Suggestion Window which uses the **F1** to **F9** keys.

How to define long forms

The text to be inserted to your word processor (i.e. the long form) should be entered in the Text to Type box of the *Dictionary Text* window. You can bring up the Dictionary Text window by either creating a new entry (p. 24) or modifying an existing one (p. 34).



Notes

- With the exception of Shorthand Tags (p. 65), Shorthand will type out whatever you enter in the Text to Type box exactly as you type it in.
- Shorthand can only simulate keystrokes. To do special things like change font styles (p. 70) or insert special characters (p. 71) you will have to use Shorthand's {@KEY} tag to simulate your word processor's *keyboard command.* Major word processors allow you to assign keyboard shortcuts to virtually any of their functions so you should be able to use Shorthand to control your word processor by simulating the appropriate keyboard command sequence.
- Be careful of extra lines at the end of your long form. Extra lines at the end of your text are invisible and will be typed out by Shorthand with the rest of your text. You can determine the number of lines in the Text to Type box by clicking on **[Options]** (above the Text to Type box) and deselecting the Word Wrap option. The number of lines in the Text to Type box will be displayed above the top right corner of the Text to Type box (p. 31).

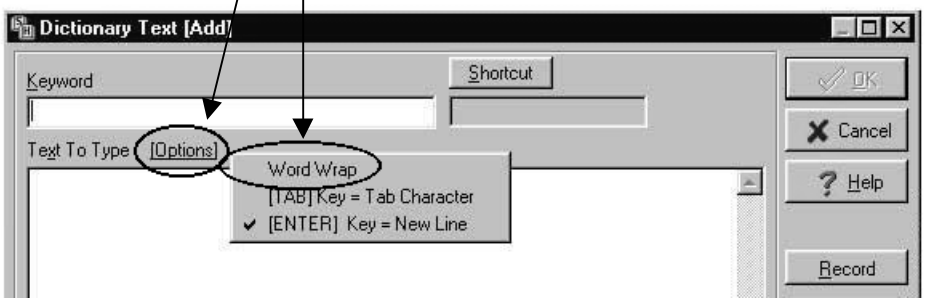
How to determine the number of lines in the Text to Type box

Shorthand types out all text in the Text to Type box including blank lines. The line counter in the Dictionary Text window helps you determine if there are blank lines at the end of your long form in the Text to Type box.

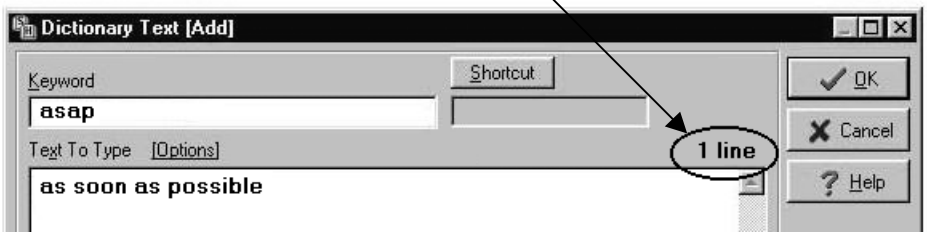
To turn on the line counter:

1 Bring up the Dictionary Text window (see p. 24 if you are creating a new entry or p. 34 if modifying an existing entry).

2 Click on **[Options]** then deselect the **Word Wrap** option by clicking on it once or twice until the checkmark disappears.



The number of lines in the Text to Type box appear just above the Text to Type box.

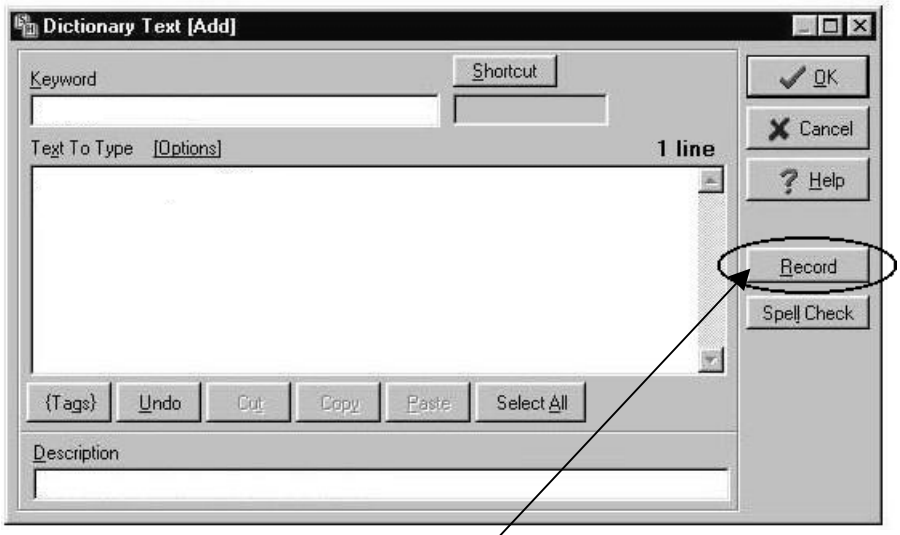


How to use the Shorthand Keystroke Recorder

To simulate special keystrokes (e.g. **F1**, **Shift+Home**, etc.) you will need to enter the appropriate {@KEY} tags in the Text to Type box. To make your job easier, Shorthand has the ability to directly record your keystrokes into the Text to Type box.

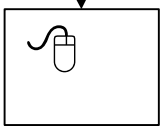
To use the recorder:

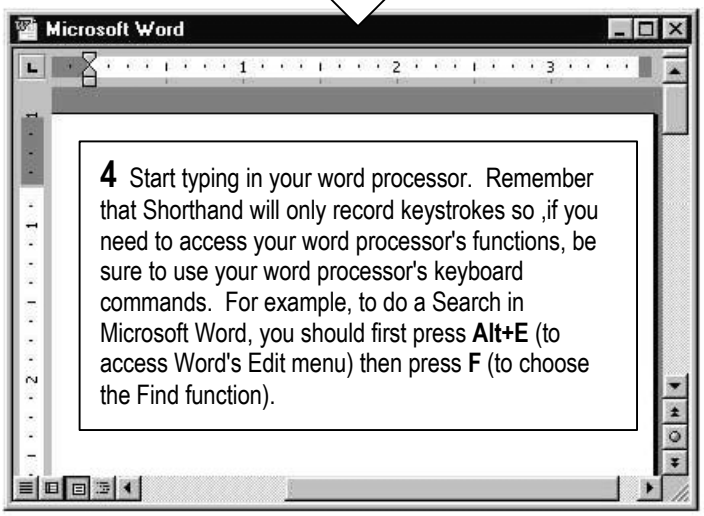
1 Bring up the Dictionary Text window (see p. 24 if you are creating a new entry or p. 34 if modifying an existing entry).



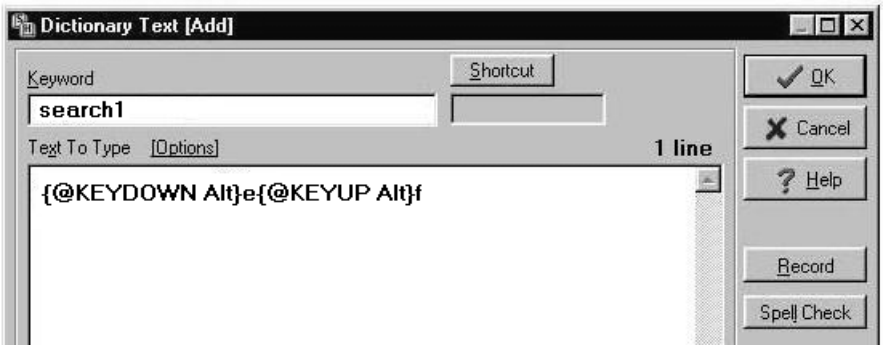
2 Click the **Record** button. The Recording Keystrokes window appears (usually at the top right corner of your screen). The title bar of the Recording Keystrokes window blinks to indicate that anything you type is now being recorded.

3 Use your mouse to switch to your word processor's window. Shorthand will not record mouse commands so anything you do with your mouse will not be recorded.





5 To stop recording, grab your mouse and click the **Stop** button in the Recording Keystrokes window. The Dictionary Text window reappears with the recorded keystrokes appended to the Text to Type box. You can edit the contents of the Text to Type box.



How to modify dictionary entries

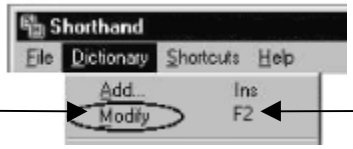
To edit or change an existing entry:

1 Bring up the Shorthand Main Window (p. 13) and select the entry you wish to modify from the Dictionary list box.

2 Use any one of the three methods below to bring up the Dictionary Text [Modify] dialog box.

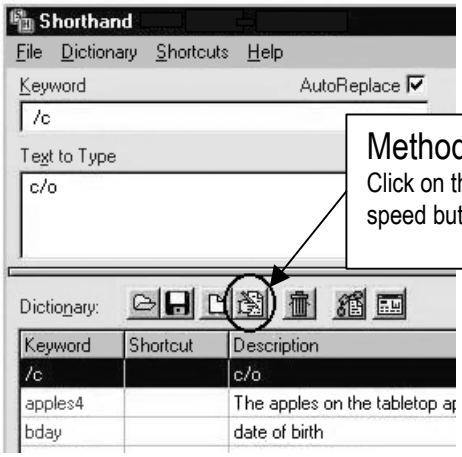
Method 1

From Shorthand's Main Menu, choose **Dictionary⇨Modify**.



Method 2

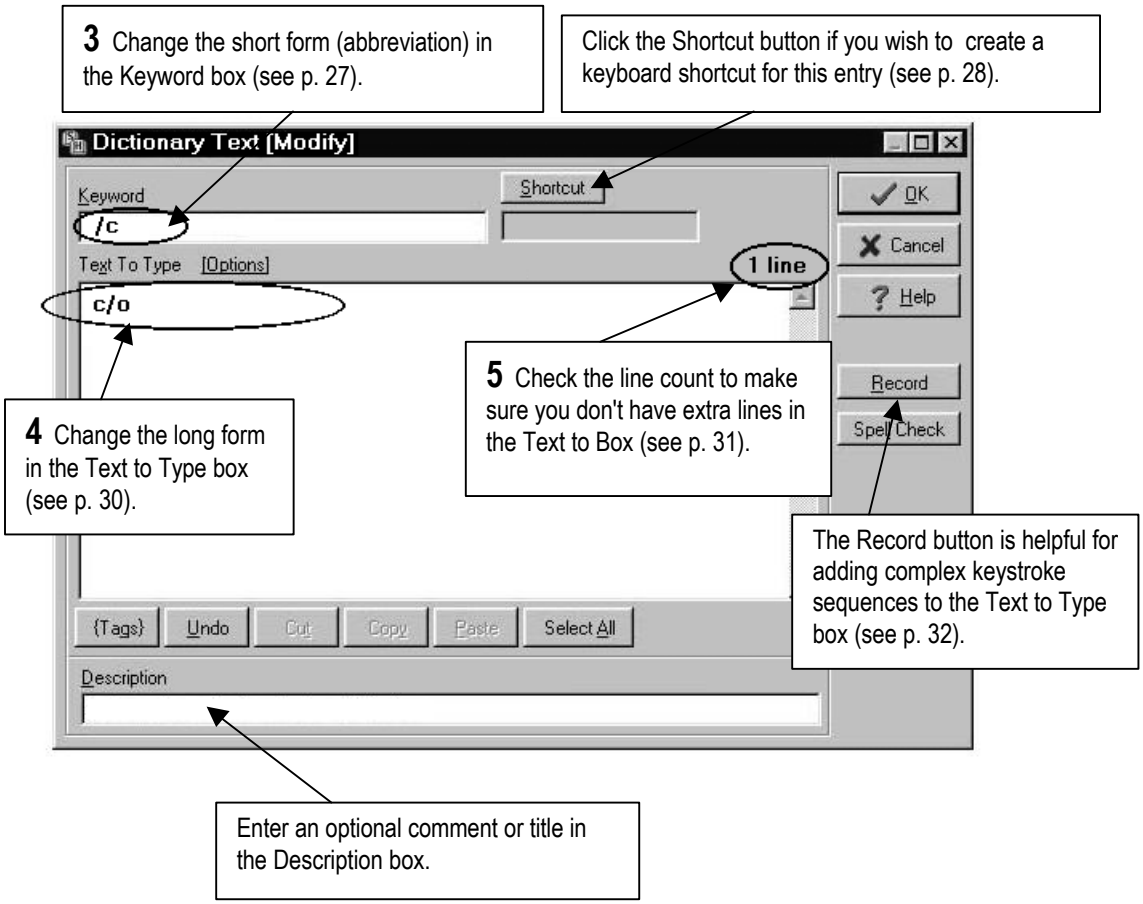
Press the **F2** key on your keyboard anywhere in Shorthand's Main Window.



Method 3

Click on the Modify speed button.

When you edit an entry, Shorthand displays the *Dictionary Text [Modify]* window for you to change the Keyword (short form) and/or Text To Type (long form):



How to determine if your dictionary has been modified.

If you modify an existing entry, create a new entry, or delete an entry Shorthand displays [modified] after the dictionary name on the title bar of the Shorthand Main Window. The [modified] disappears when the dictionary is saved.

How to duplicate an entry

You can create a new entry based on an existing entry as follows:

1 Bring up the Shorthand Main Window (p. 13) and select the entry you wish to duplicate from the Dictionary list box.

2 Use any one of the two methods below to create a new copy of the selected entry.

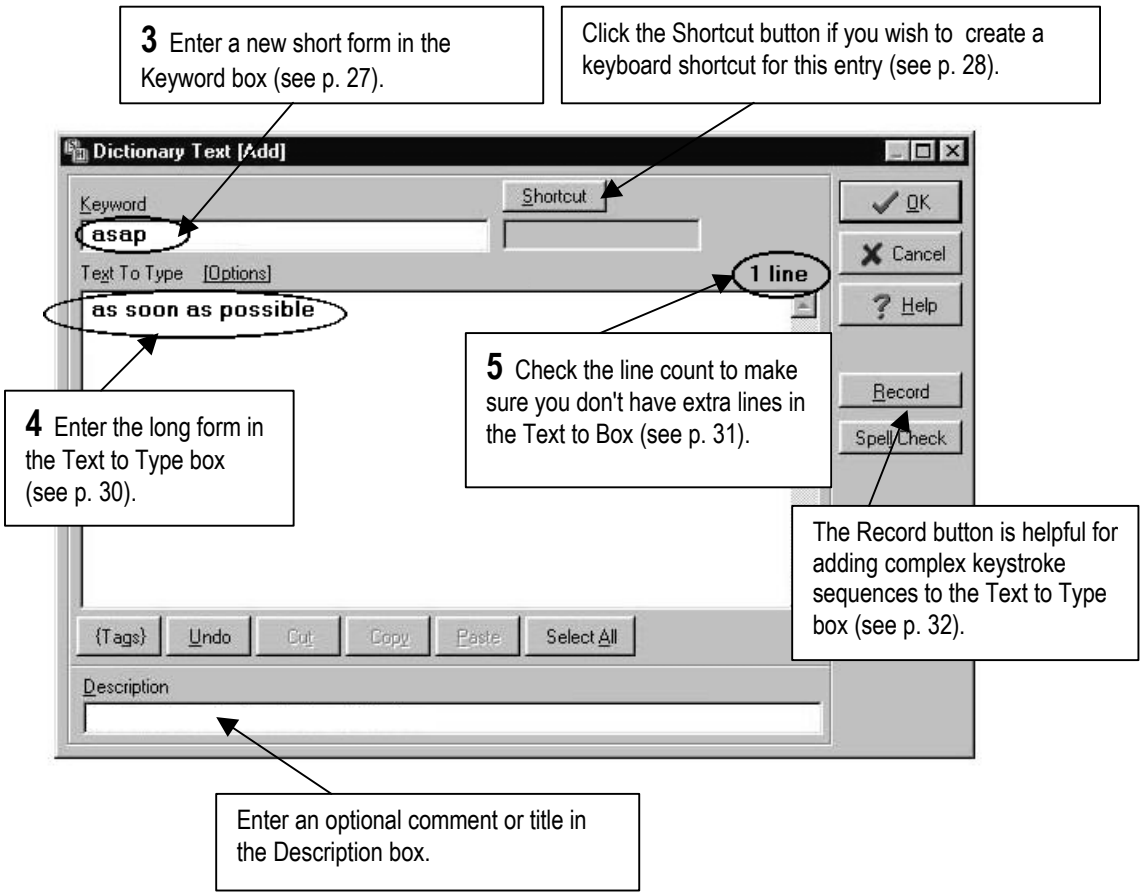
Method 1

From Shorthand's Main Menu, choose **Dictionary⇨Duplicate**.

Method 2

Press the **Alt+Ins** key on your keyboard in Shorthand's Main Window.

When you duplicate an entry, Shorthand displays the *Dictionary Text [Add]* window for you to enter a new Keyword (short form) and Text To Type (long form). Remember that you are editing a *copy* of the original entry so you should change the keyword to something else as Shorthand does not allow two entries to have the same keyword.

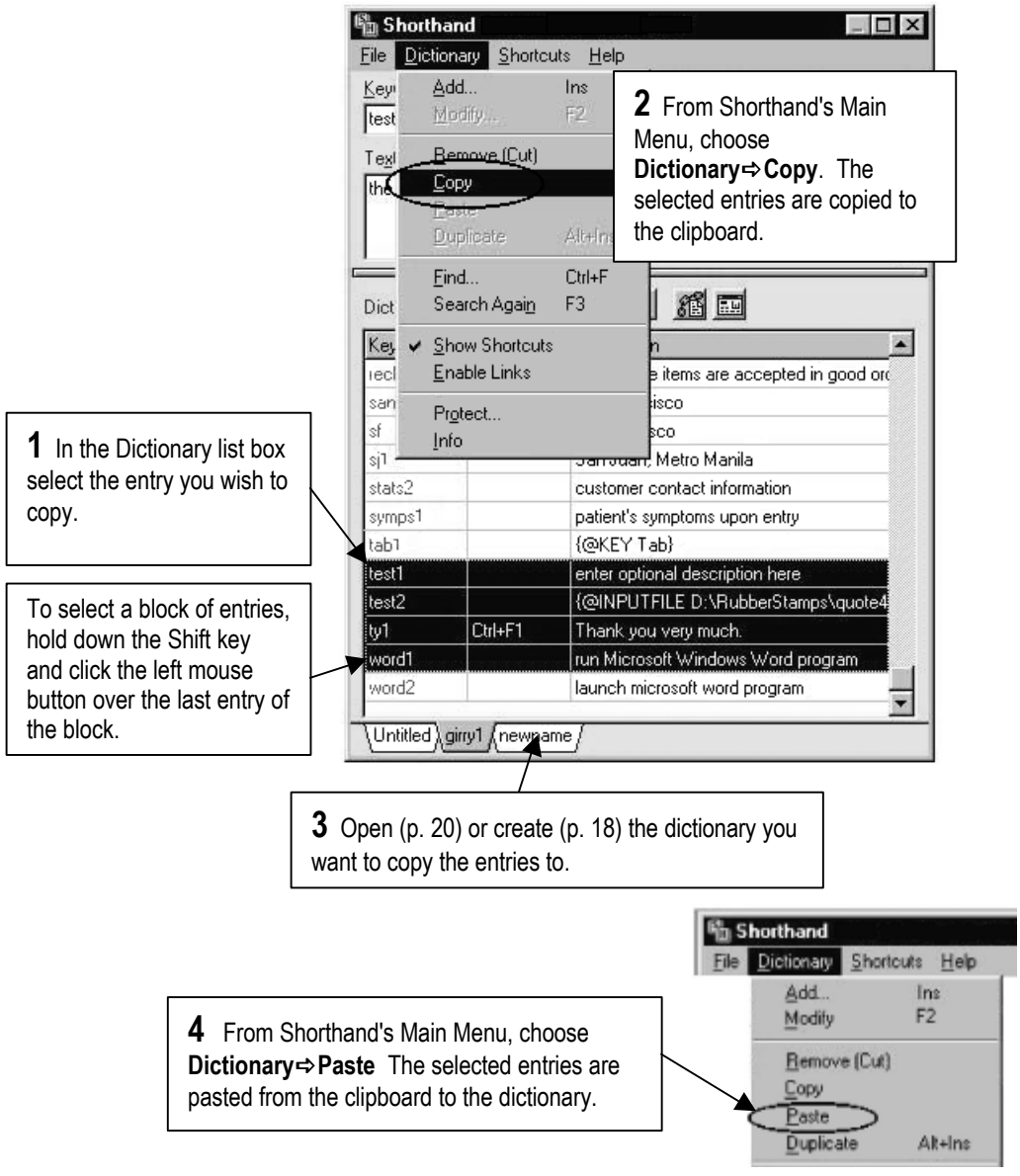


Use the Duplicate function to quickly create different forms of an abbreviation.

For example, if you have an entry for immunize , you can quickly create a new abbreviations for immunization , immunizing , etc. by duplicating and modifying the immunize entry.

How to copy an entry to another dictionary

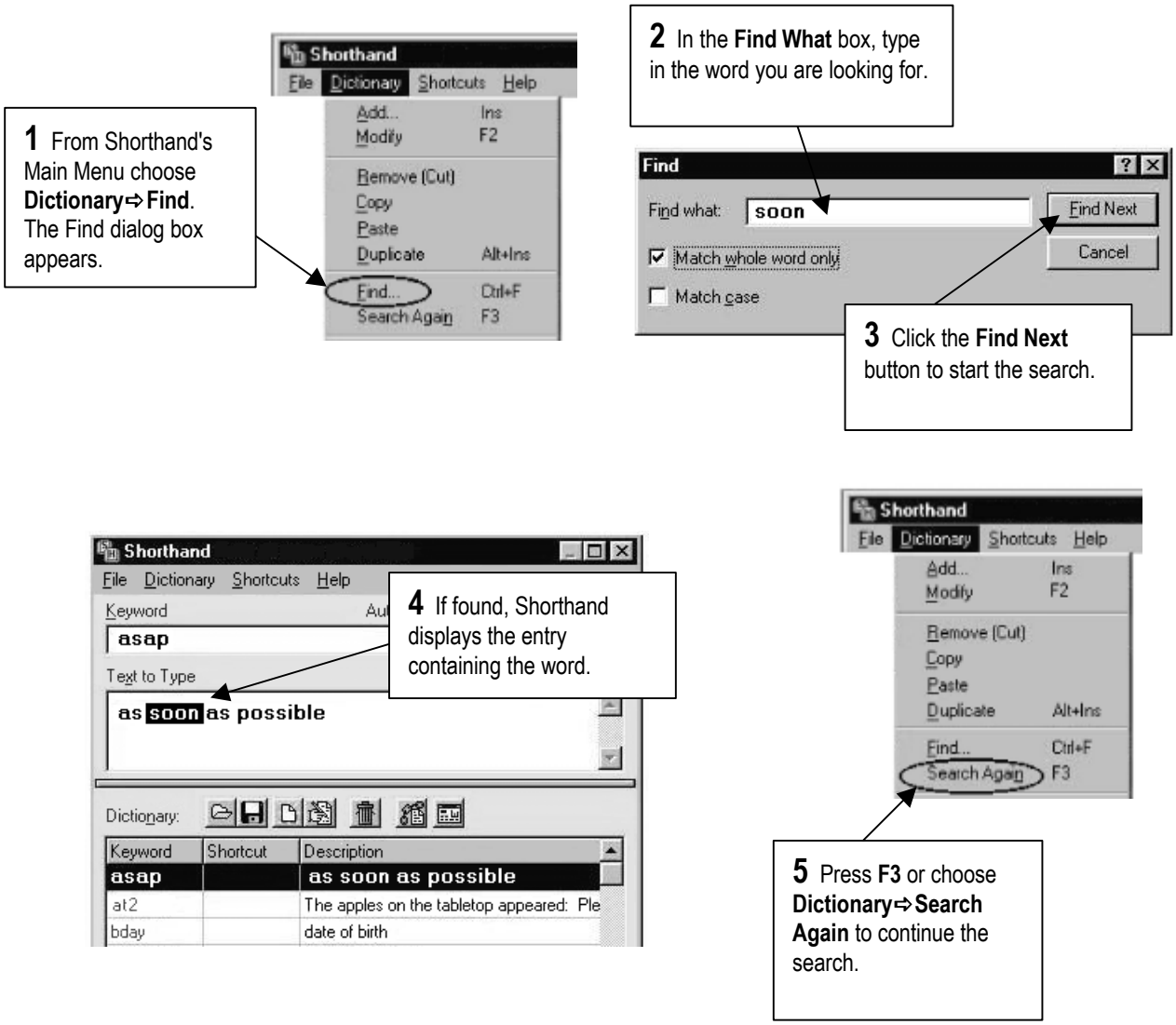
To copy dictionary entries to another dictionary:



☞ If a pasted entry already exists, Shorthand automatically assigns a unique keyword (the duplicate keyword will have a number after it).

How to search through dictionary entries

If you are looking for a dictionary entry that contains a particular word, you can perform a simple search through the active dictionary as follows:



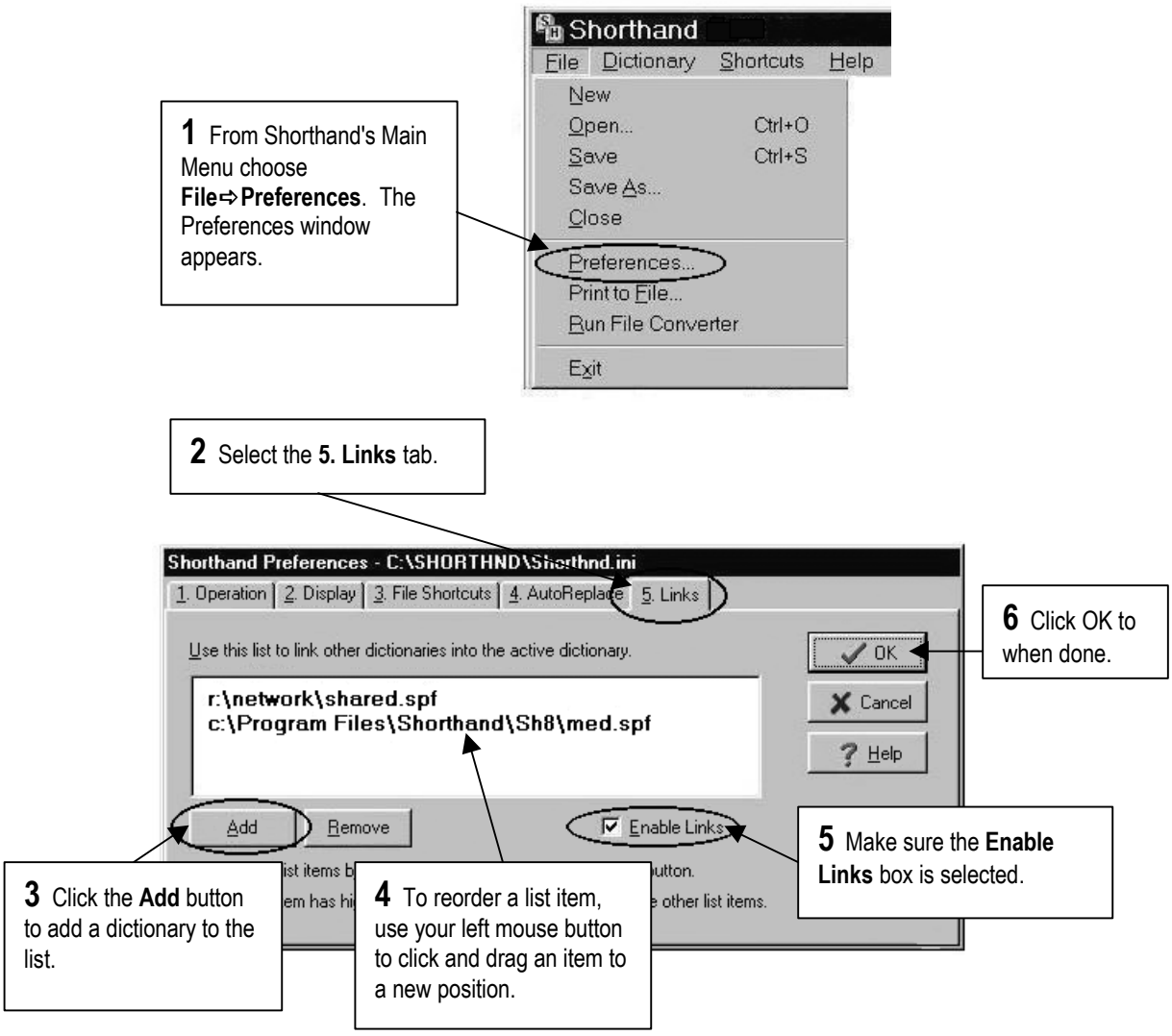
- The **Dictionary⇨Find** function searches through the Text to Type and Description fields; to search for a Keyword, simply type in the Keyword box (p. 15)
- See also p. 122 for a Tcl script that performs searches with wild card pattern matching.

How to link dictionaries

Shorthand allows only one active dictionary at a time. However, you can *link* inactive dictionaries to the active dictionary which allows for access to more than one dictionary at a time.

Adding dictionaries to the Links list

Linking a dictionary is simply a matter of specifying the dictionary's filename in Shorthand's Links list. To add a dictionary to Shorthand's Links list:



How linking works

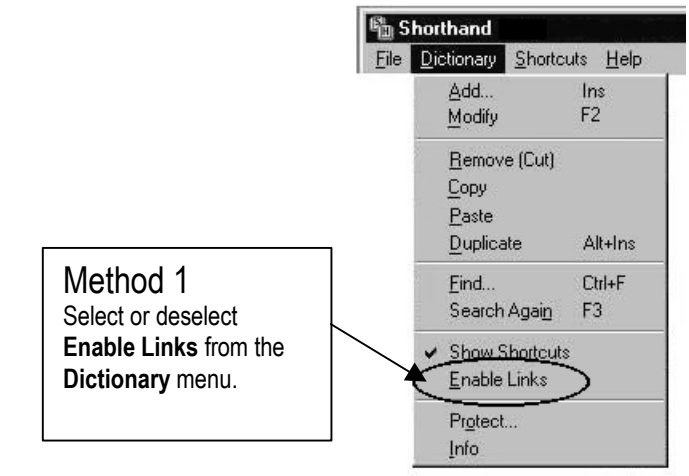
When you open a dictionary (e.g. by choosing **File⇨Open)** Shorthand first reads in the entries of the dictionary you specify and, if the Enable Links option is enabled, Shorthand proceeds to read in the entries of the dictionaries in the Links list.

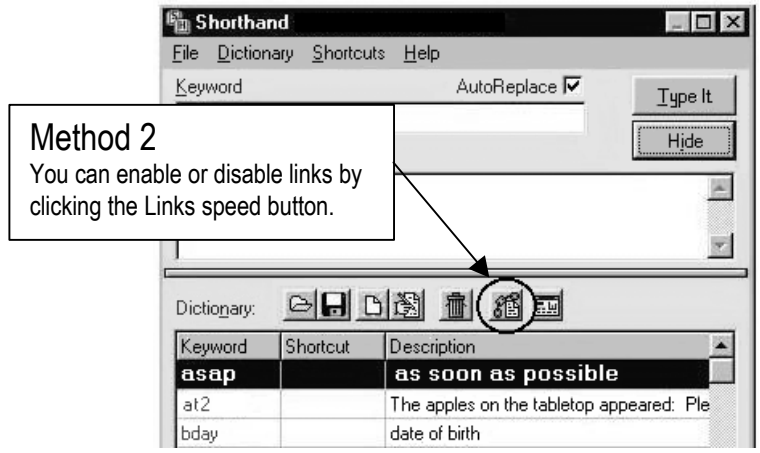
The sequence of the dictionaries in the Links list box is important as Shorthand will load the linked dictionaries in the order they appear in the list. Shorthand doesn't support multiple entries with the same keyword so, if the same keyword appears in two or more linked dictionaries, the keyword in the dictionary that is loaded first gets priority.

☞ You can only edit entries in the active dictionary (i.e. the dictionary you opened last and whose name appears when you choose **Dictionary⇨Info**). Linked entries cannot be modified. To modify a linked entry, you will first have to make its dictionary the active dictionary by opening it (see p. 20).

How to enable/disable links

To unlink (unmerge) dictionaries, deselect the Enable Links option. To link the dictionaries again, select the Enable Links option.





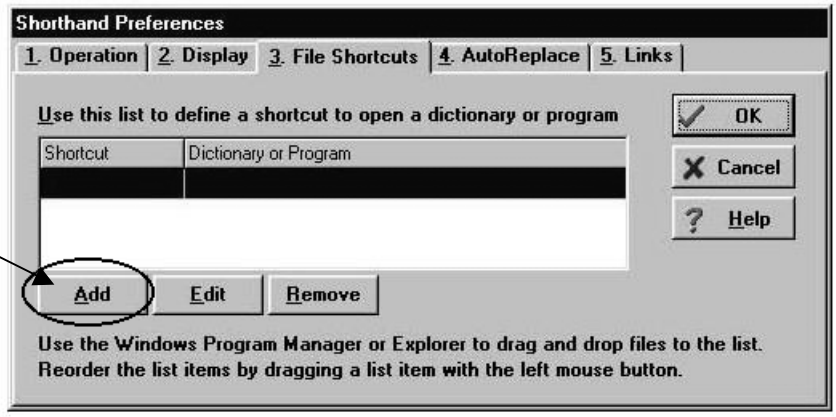
How to create a File Shortcut

You can create a *file shortcut* to open a Shorthand dictionary, launch an external application (e.g. Microsoft Word) or run a Shorthand Tcl script with a single keystroke.

To create a file shortcut:

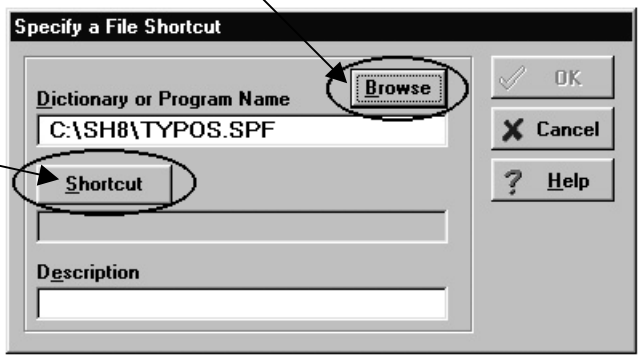
1 From Shorthand's Main Menu choose **Shortcuts⇨Edit**. The **3. File Shortcuts** tab of the Shorthand Preferences window appears.

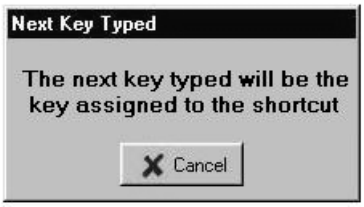
2 Click the **Add** button. The Specify a File Shortcut dialog box appears.



3 Click the **Browse** button and select the file to be associated with the shortcut key.

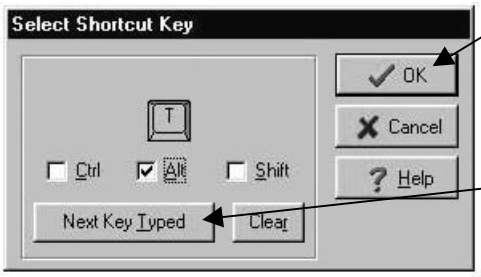
4 Click the **Shortcut** button. The Next Key Typed window appears.





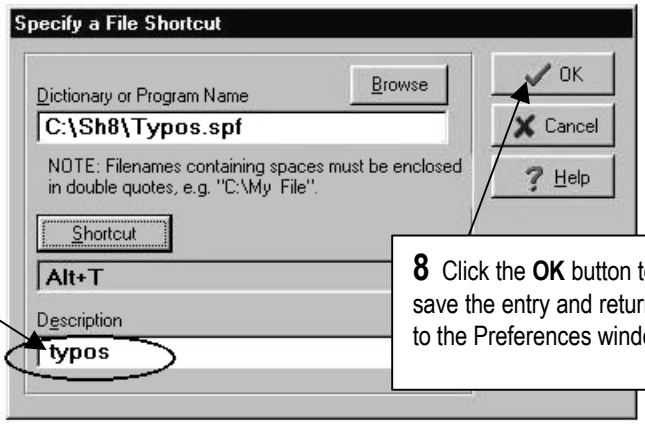
5 In the Next Key Typed window, press the keystroke (e.g. **Alt+T**) you want assigned as the shortcut. The Select Shortcut Key window appears and shows the key you just typed.

6 Click the **OK** button to accept the keystroke. The Specify a File Shortcut window reappears.

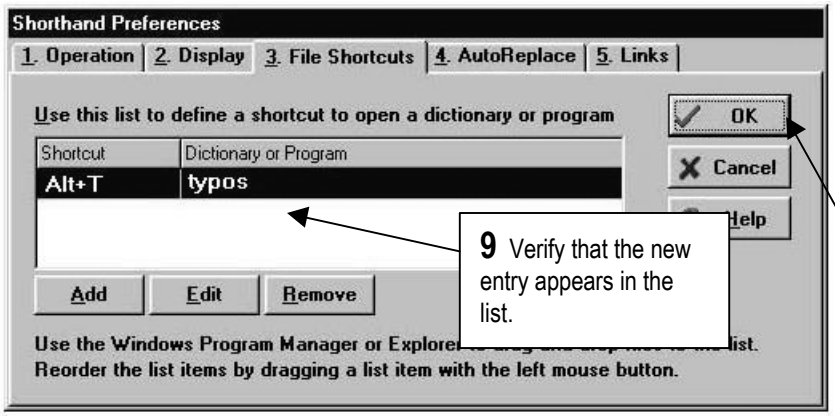


If you want to change the keystroke, click the Next Key Typed button to start over.

7 In the Description box, enter a short string describing the command.



8 Click the **OK** button to save the entry and return to the Preferences window.

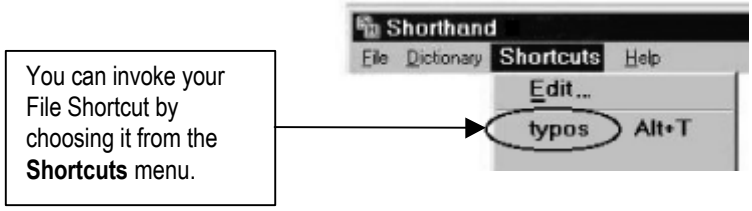


9 Verify that the new entry appears in the list.

10 Click the **OK** button to close the Preferences window and return to Shorthand.

Using the File Shortcut

You can now open or launch the specified file by pressing the shortcut key (e.g. **Alt+T**) from within your word processor. You can also invoke the file shortcut within Shorthand by choosing Shortcuts from Shorthand's Main Menu.



Notes

- If you want to use function keys as shortcut keys it is recommended that you combine them with the **Ctrl**, **Shift**, and/or **Alt** key to avoid conflicts with the Suggestion Window which uses the **F1** to **F9** keys.
- You can specify almost any type of file to be associated with a File Shortcut. For example if you associate **Alt+W** with the file "http://www.pcshorthand.com", pressing **Alt+W** will launch your web browser and go to the specified web page.

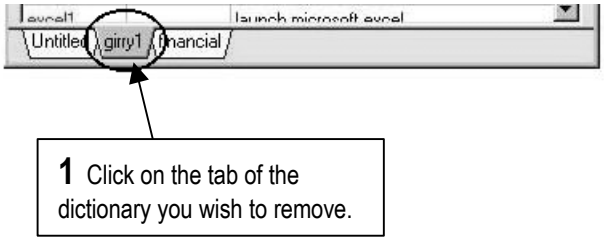
File Shortcuts and Tcl scripts are useful for adding new functionality to Shorthand.

You can add new menu commands to Shorthand by writing a Tcl script and creating a File Shortcut to run the script. For example, see p. 122 for the procedure to add a menu command to perform a search with pattern matching.

How to remove a dictionary tab

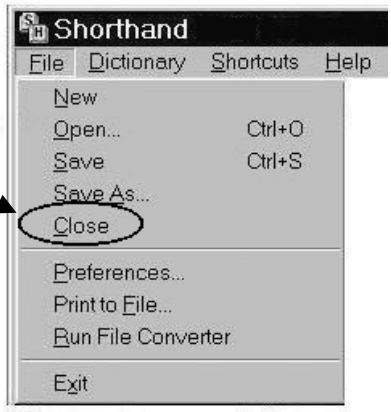
When you open or create a dictionary, Shorthand creates a tab for it at the bottom of the Main Window. If you don't want to see a tab for a certain dictionary, you can remove the tab by closing the dictionary.

To close a dictionary:



1 Click on the tab of the dictionary you wish to remove.

2 Choose **File⇨Close**. The tab disappears and file in the next tab is opened.



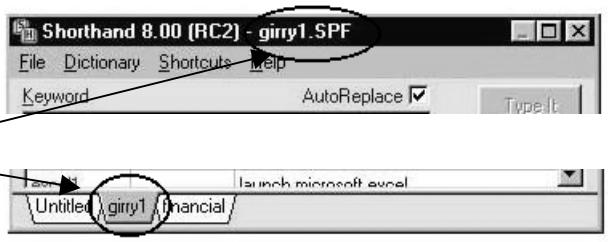
☞ When you close a dictionary, you are simply removing its tab from the Main Window; the file has *not* been deleted. To permanently remove a file from your hard disk, you will need to use Windows Explorer (p. 11) to delete the file.

How to protect a dictionary from unauthorized access

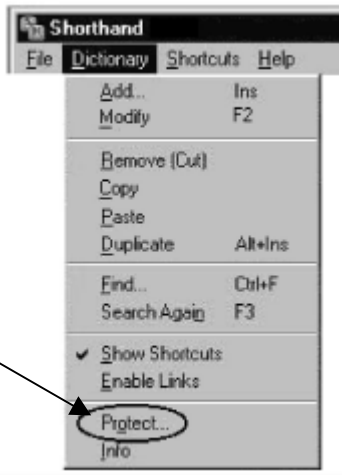
Shorthand gives you the option of protecting dictionary files from unauthorized use. This is especially helpful if you plan on sharing your dictionaries with another user but want to prevent others from accessing the file. Protecting a dictionary file also prevents changes to its contents.

To protect a dictionary:

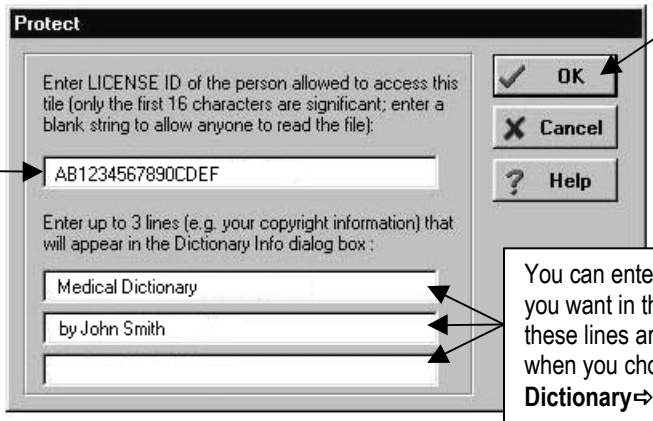
1 Open the dictionary you want to rename (p. 20). The dictionary's name should appear in Shorthand's title bar and be the selected dictionary tab.



2 From Shorthand's Main Menu, choose **Dictionary⇨Protect**. The Protect dialog box appears.



3 Enter the License ID (p. 47) of the person who may access the dictionary. If you enter your own License ID, then only computers registered with your Shorthand license will be able to access the dictionary. If you enter nothing in the first box, anyone with Shorthand can access the dictionary (i.e. the dictionary is unprotected).



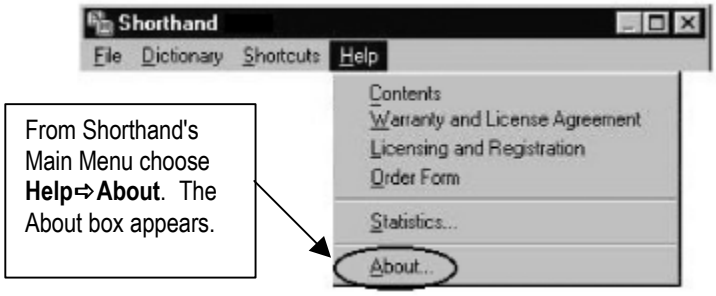
4 Click OK to apply the protection to the active dictionary

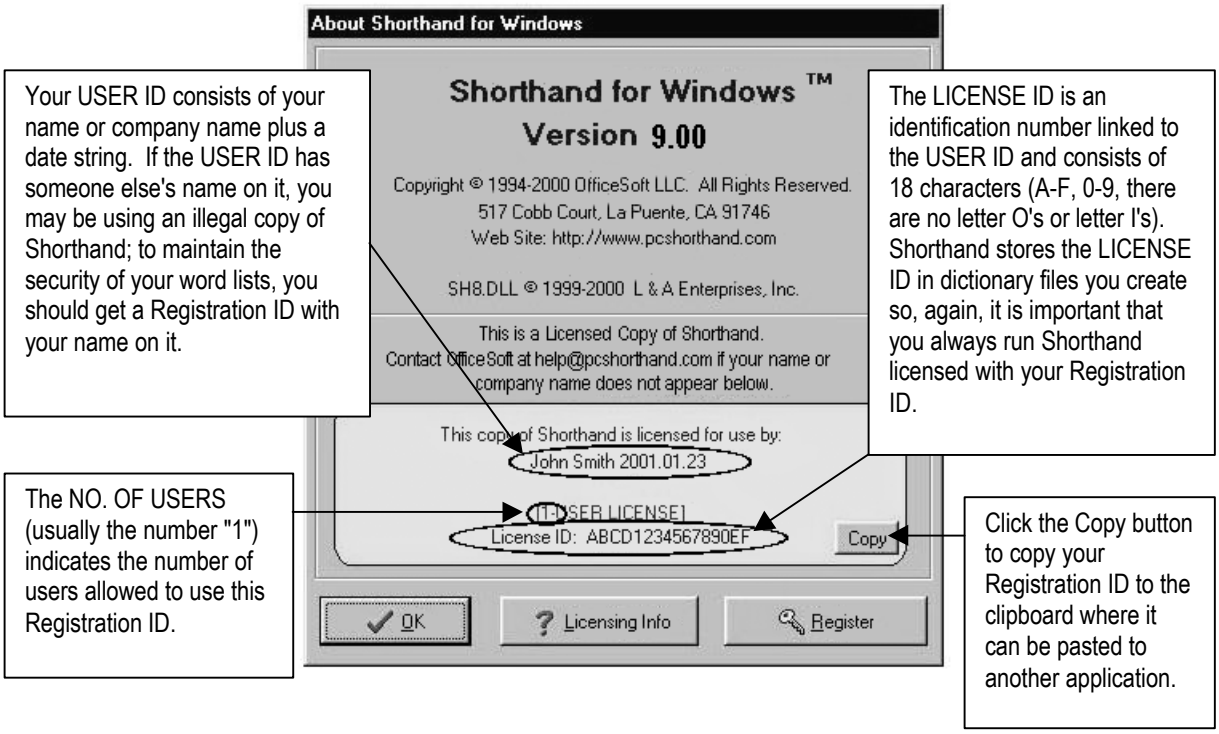
You can enter anything you want in these 3 lines; these lines are displayed when you choose **Dictionary⇨Info.**

Where is my Registration ID?

When you purchased a Shorthand License, you should have received a personalized *Registration ID* with instructions on how to enter it into your copy of Shorthand. Your Registration ID has 3 parts: the USER ID, the NUMBER OF USERS and the LICENSE ID.

To view your Registration ID information:





How to open a dictionary with a single keystroke

You can open a dictionary with a single keystroke by creating a File Shortcut for that dictionary. Follow the procedure on p. 42 to create a file shortcut and specify the dictionary file to open in step 3.

How to launch external applications with a single keystroke

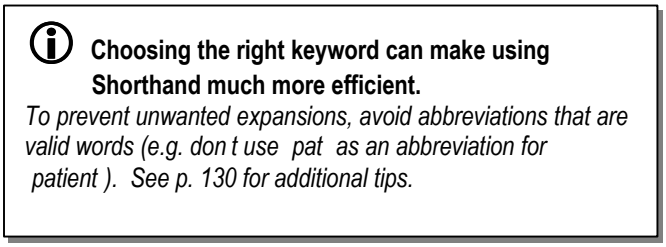
You can launch a Windows application with a single keystroke by creating a File Shortcut. Follow the procedure on p. 42 to create a file shortcut and specify the filename of the Windows application to launch in step 3.

How to prevent a keyword from expanding

Press **Esc** right after you type the keyword.

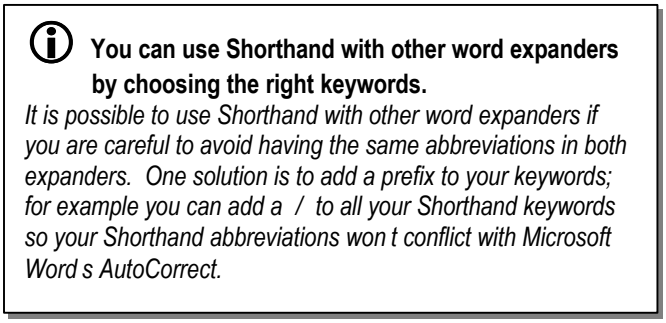
How to replay an expansion

If, for some reason, an expansion was not done correctly you can replay the last expansion by pressing the Shorthand Hot Key (p. 49) in your word processor to bring up Shorthand's Main Window, right click on the Text to Type box, select "Restore Last" then click the Type It button.



Choosing the right keyword can make using Shorthand much more efficient.

To prevent unwanted expansions, avoid abbreviations that are valid words (e.g. don t use pat as an abbreviation for patient). See p. 130 for additional tips.



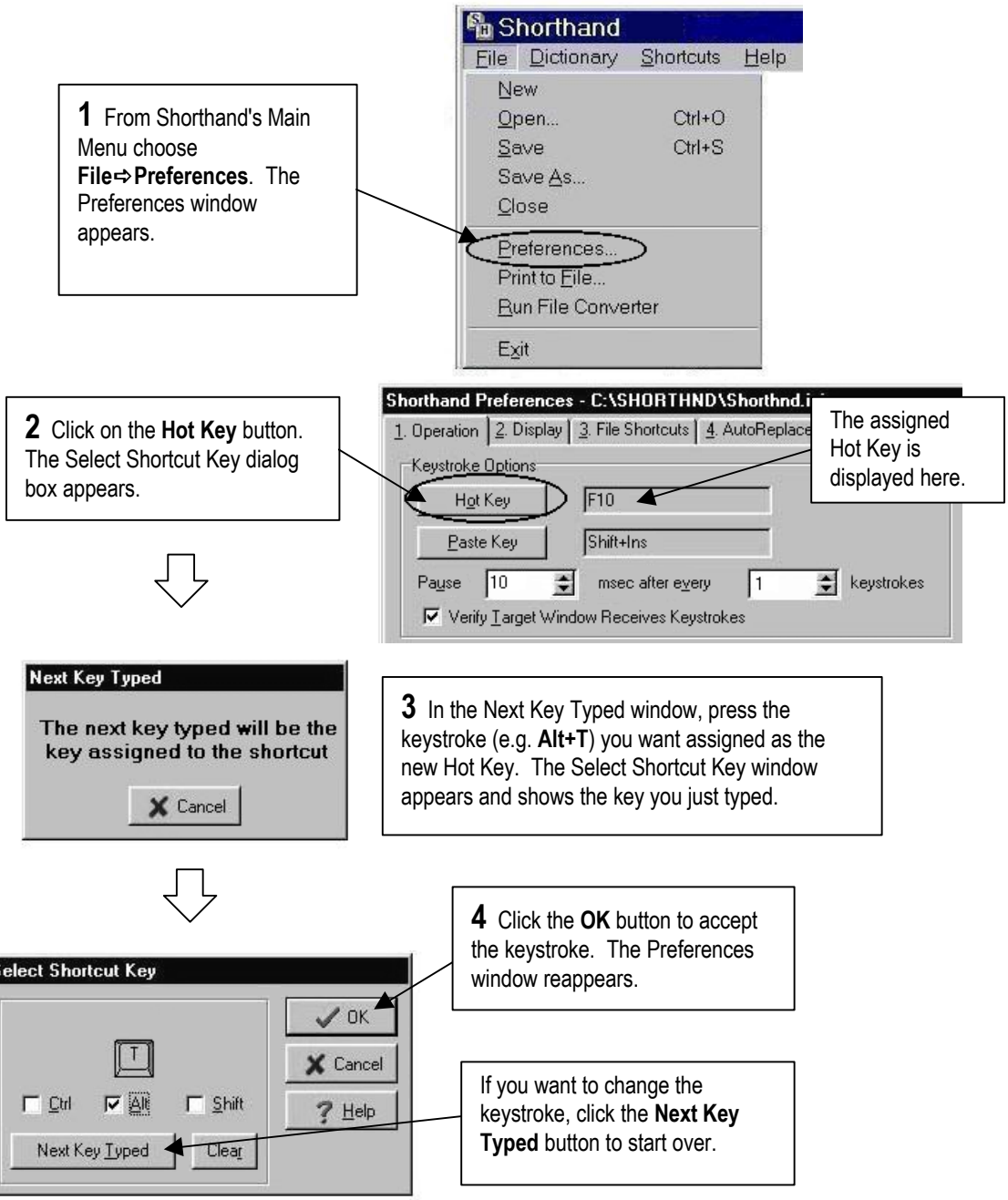
You can use Shorthand with other word expanders by choosing the right keywords.

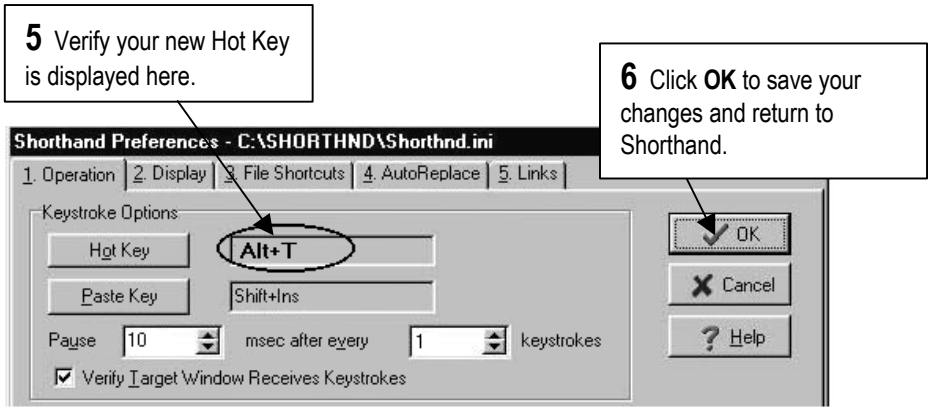
It is possible to use Shorthand with other word expanders if you are careful to avoid having the same abbreviations in both expanders. One solution is to add a prefix to your keywords; for example you can add a / to all your Shorthand keywords so your Shorthand abbreviations won t conflict with Microsoft Word s AutoCorrect.

Chapter 3 Configuring Shorthand

Hot Key

The *Hot Key* is the keyboard command to access the Shorthand Main Window. By default, Shorthand assigns **F10** as the Hot Key. To change the Hot Key:





Notes

- Do not use function keys **F1** to **F9** as keyword shortcut keys as it may conflict with the Suggestion Window which (unless you change this) uses the **F1** to **F9** keys to select a suggestion.
- The Hot Key only works if the Shorthand program is running and the Shorthand icon appears in the system tray (p. 13).
- The Hot Key is not the only way to bring up Shorthand's Main Window. Alternative ways of bringing up the Main Window are clicking on the Shorthand system tray icon (p. 13) or clicking your right mouse button over the Suggestion Window (p. 56). Note that the Type It button is enabled only if you bring up the Main Window with the Hot Key. The Type It button is disabled if you click on the Shorthand icon or system tray icon because Shorthand doesn't know to which window to perform the expansion.

AutoReplace

When *AutoReplace* is turned on, Shorthand will automatically detect and replace keywords as you type in your word processor. Since this is usually what you want Shorthand to do, always make sure AutoReplace is enabled before using Shorthand.

To make sure AutoReplace is activated:

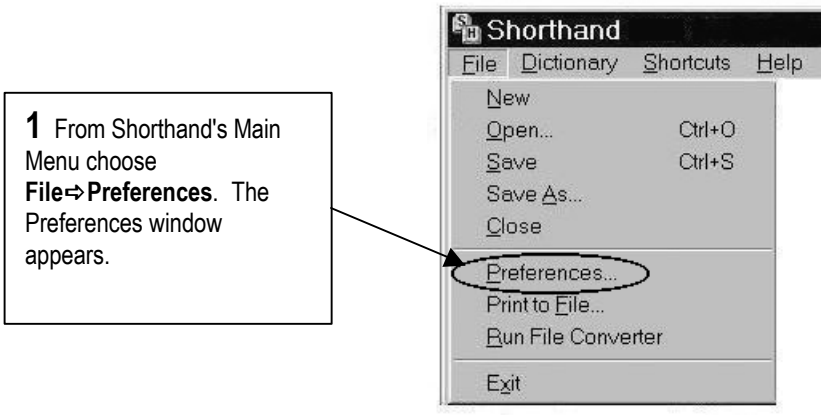
1 Bring up Shorthand's Main Window (p. 13).

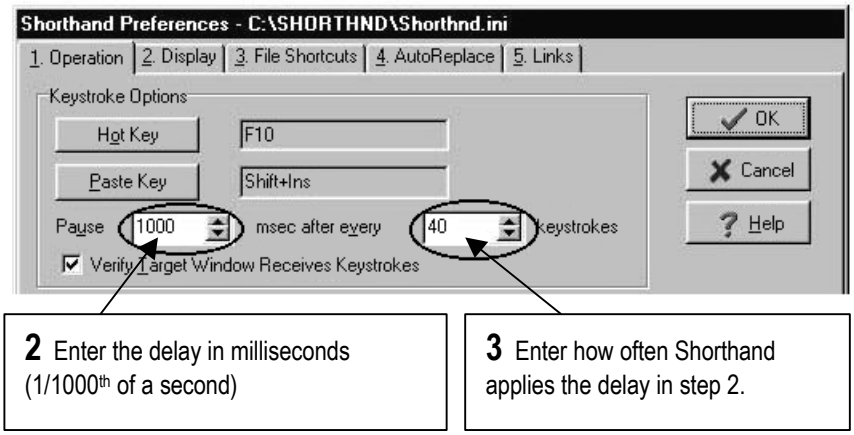
2 Make sure the **AutoReplace** box is selected.

Expansion speed

Word processors are designed to accept input from human typists. A common problem is when Shorthand inserts text far too fast for your word processor to process which can result in doubled first characters, missing characters and parTIAL caps. If this occurs, the solution is to slow down Shorthand to give your word processor time to keep up with Shorthand.

To change Shorthand's typing speed:





Notes

- A setting of **100 msec every 3 keystrokes** means Shorthand will pause 100 milliseconds (100/1000th of a second) after every third keystroke it inserts.
- A setting of **20 msec every 1 keystrokes** should work with most newer computers. If you observe missing characters in the expansion, try increasing the delay to **50 msec** (or higher) every **1 keystrokes**.
- If changing the delay doesn't seem to help, another application may be interfering with Shorthand. Try identifying and shutting down all programs that monitor keystrokes; examples are spell checkers, clipboard utilities, mouse utilities (e.g. Power Toys) and built-in word expanders such AutoCorrect in Microsoft Word.
- Setting the delay to **0 msec** means Shorthand will simulate keystrokes at full speed and is *not* recommended.

Use the {@PAUSE} tag to insert delays when expanding large amounts of text.

Most word processors need to redraw its window when text scrolls off the bottom of the page. For this reason, you should embed {@PAUSE} *tags (p. 72) in long forms with a large amounts of text.*

Other programs can affect Shorthand s expansion speed.

When Shorthand pauses, Shorthand will not wake up until other applications have completed their activity. Unless you have a fast computer, it is recommended you do not run any application that works in the background; examples are: timers, automatic email or news notifiers, mouse/keyboard utilities, the Microsoft Office Toolbar and spell checkers.

Text transfer methods

Shorthand provides two methods for inserting text to your word processor: *Simulate Keystrokes* and *Simulate Clipboard Paste*.

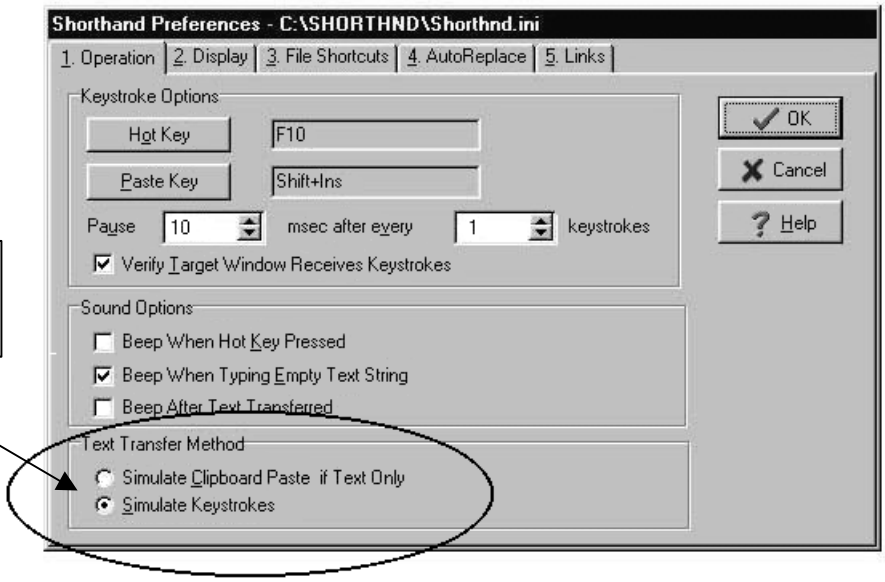
The *Simulate Keystrokes* method (the default method) synthesizes a full keystroke for every character that needs to be inserted to your word processor. This method works well with most applications.

The *Simulate Clipboard* method first transfers the Shorthand text to the clipboard and then simulates a **Shift+Ins** keystroke to paste the text from the clipboard to your word processor; use this method if you usually need to transfer large amounts of text. Note that Simulate Clipboard works if only pure text are to be inserted; if your Shorthand code contains {@KEY} or {@PAUSE} tags, Shorthand will use the Simulate Keystrokes method.

To select the text transfer method

1 From Shorthand's Main Menu choose **File⇨Preferences**. The Preferences window appears.

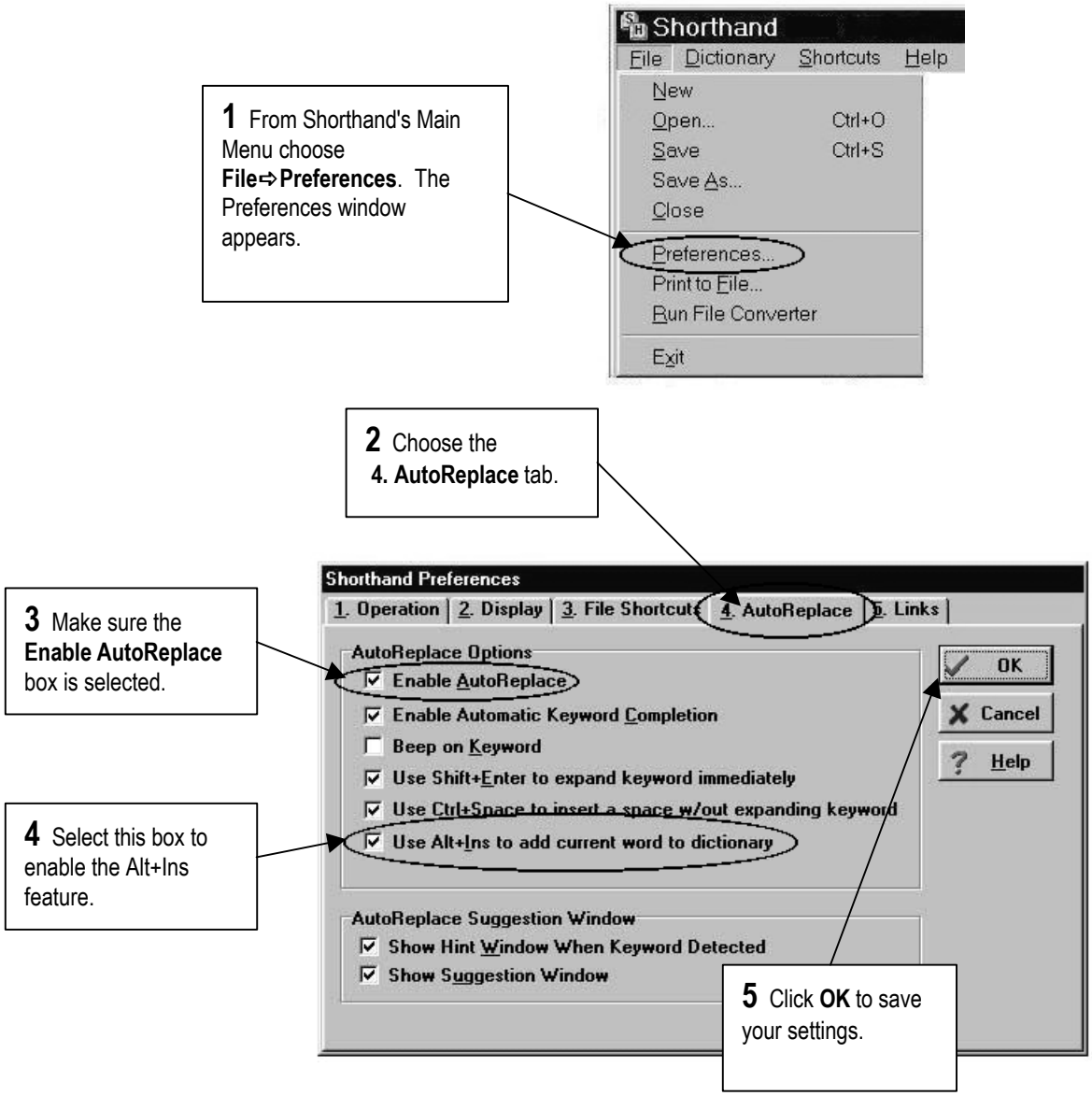
2 Select the Text Transfer method.



Adding keywords with the Alt+Ins command

Shorthand allows you to quickly add new keywords to the active dictionary by pressing **Alt+Ins** from your word processor.

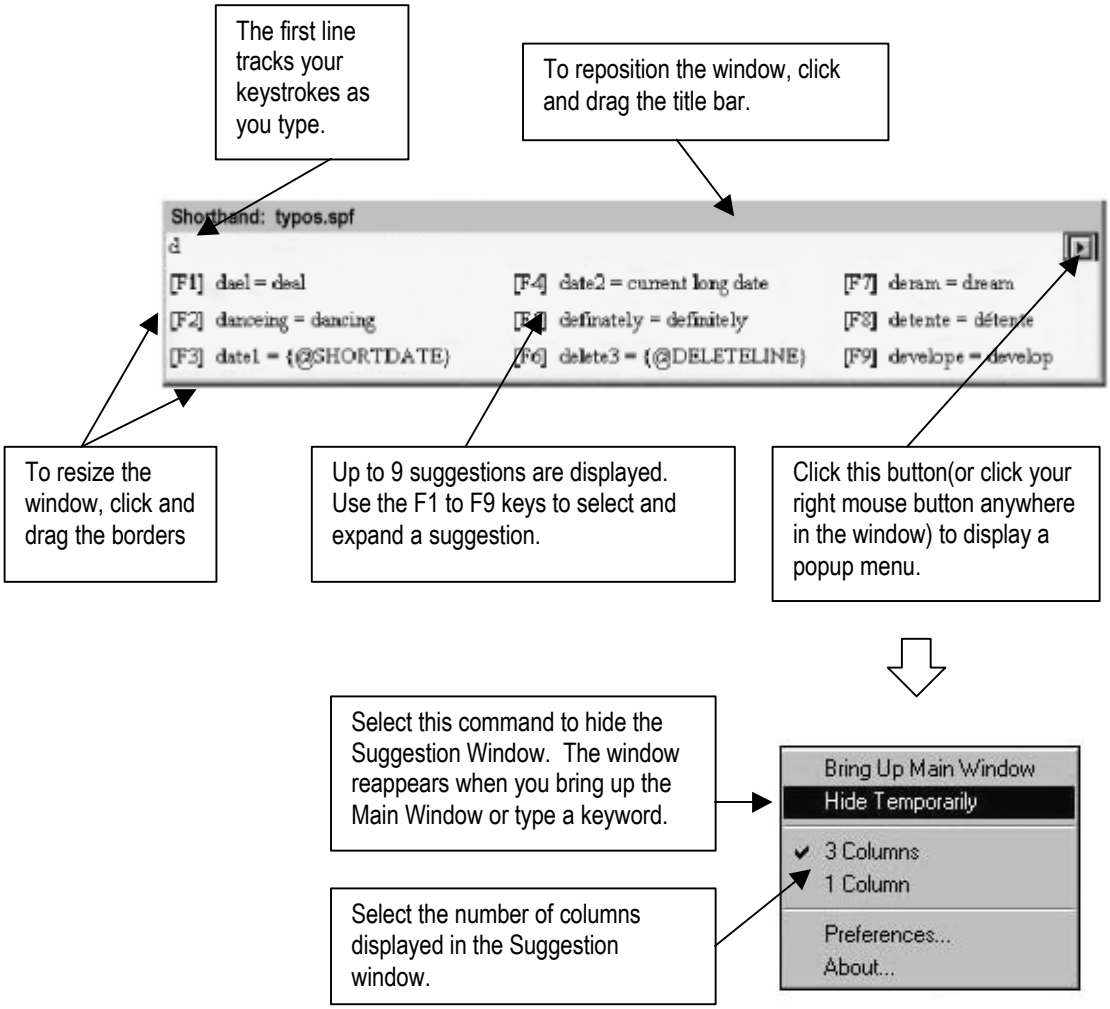
To enable the Alt+Ins feature:



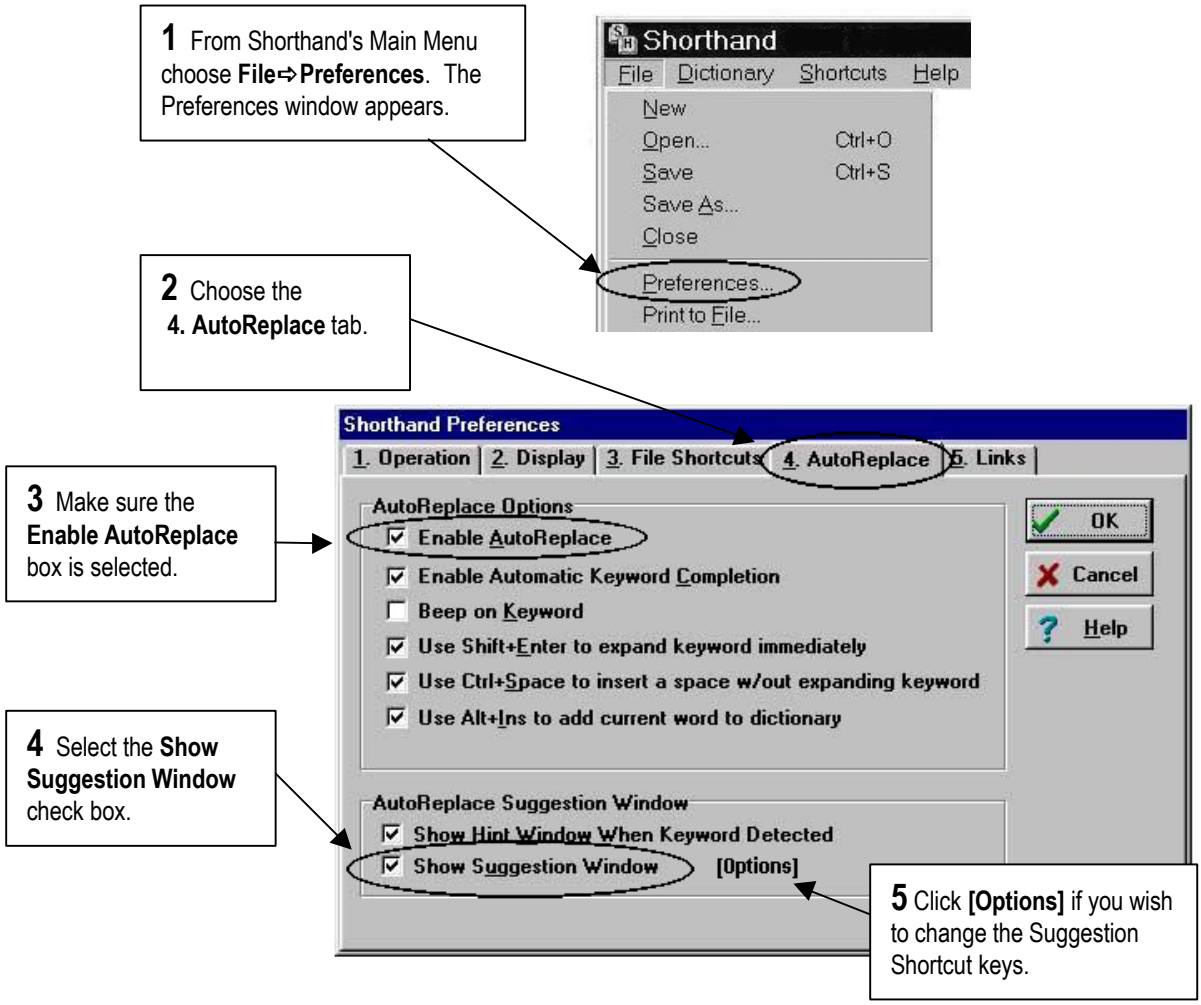
☞ The **Alt+Ins** command works only if AutoReplace (p. 51) is enabled.

Suggestion Window

The *Suggestion Window* displays a list of up to nine suggestions for the word you just typed. You can expand an entry by selecting from the list with the **F1** to **F9** keys.



To enable the Suggestion Window:



☞ Shorthand displays the Suggestion Window only if AutoReplace (p. 51) is enabled.

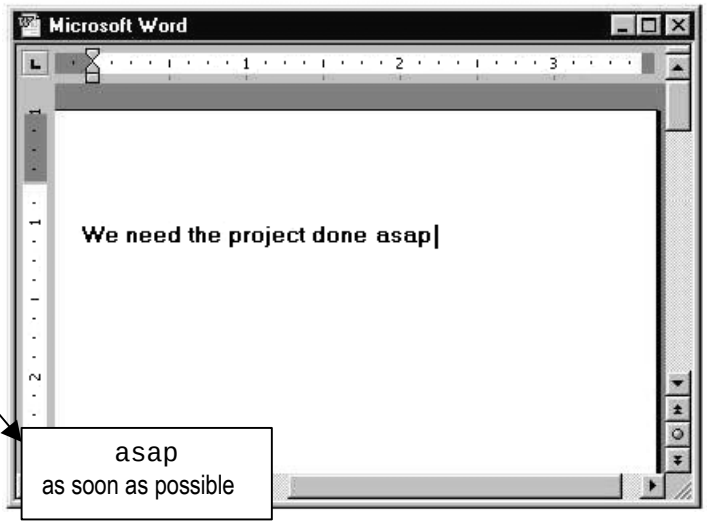
ⓘ **You can change the size of the text displayed in the Suggestion Window.**

If the text in the Suggestion Window appears too small or too large, you can change the fonts as well as the background window color. See p.61 on how to do this.

Hint Window

To enable the Hint Window:

Shorthand can display a small *Hint Window* whenever you type a recognized keyword. The Hint Window displays the associated text that will be inserted in your word processor if you press ENTER , SPACE or any punctuation mark. To prevent expansion, press ESC .

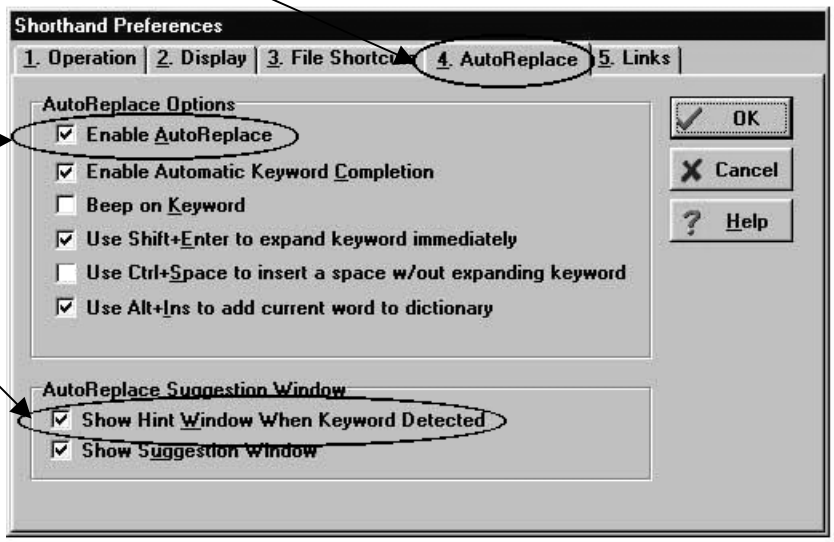


1 From Shorthand's Main Menu choose **File⇨Preferences**. The Preferences window appears.

2 Choose the **4. AutoReplace** tab.

3 Make sure the **Enable AutoReplace** box is selected.

4 Select the **Show Hint Window When Keyword Detected** check box



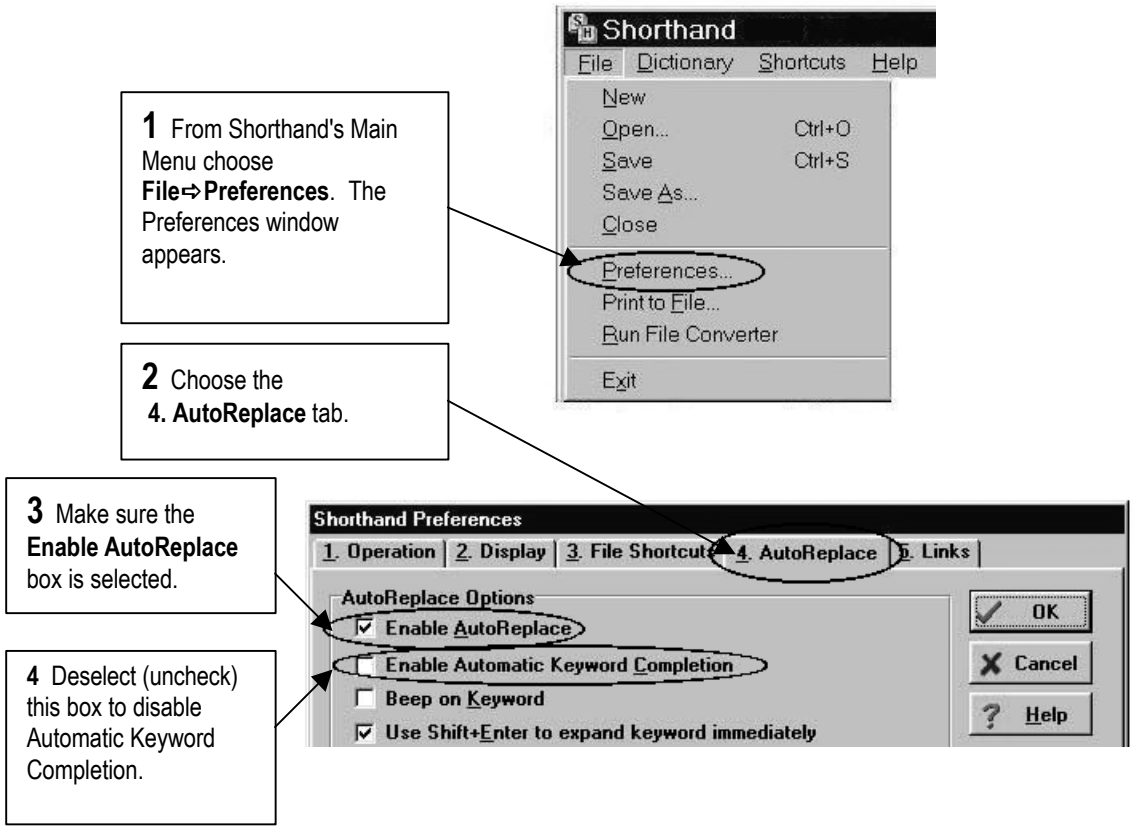
Automatic Keyword Completion

Automatic Keyword Completion is helpful if you have trouble remembering keywords. When Automatic Keyword Completion is enabled (the default), Shorthand will display the nearest matching keyword whenever you type two or more commas after a word. For example if you have the following keywords:

Keyword	Text to Type
`ty1`	`Thank you very much`
`ty2`	`Thank you`
`ty3`	`Thanks`

If you type `ty,,` ("ty" followed by two commas), Shorthand's *Hint Window* (p. 58) and *Suggestion Window* (p. 56) will display the first keyword that begins with "ty" which in our example is the keyword for "Thank you very much." If you type `ty,,,` ("ty" followed by three commas), Shorthand will match this to `ty2`, the keyword for "Thank you." Similarly, `ty,,,,` (the word "t" followed by four commas) will be matched to the keyword `ty3` which is the keyword for "Thanks".

By default, Automatic Keyword Completion is turned on; to disable it:

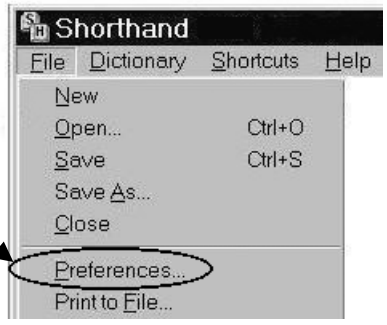


Sound options

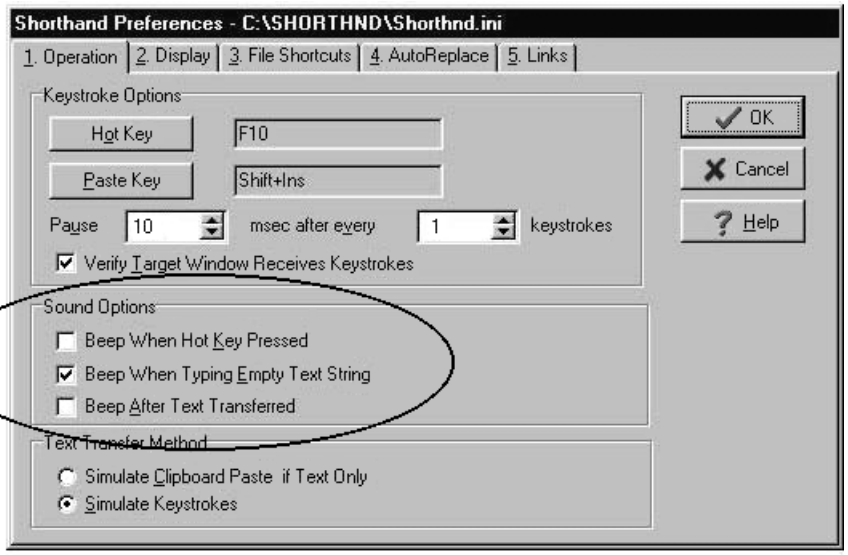
For touch typists, Shorthand can give audible feedback on what it is doing so you don't have to look at the screen.

To access Sound Options:

1 From Shorthand's Main Menu choose **File⇨Preferences**. The Preferences window appears.



2 Select the appropriate options.



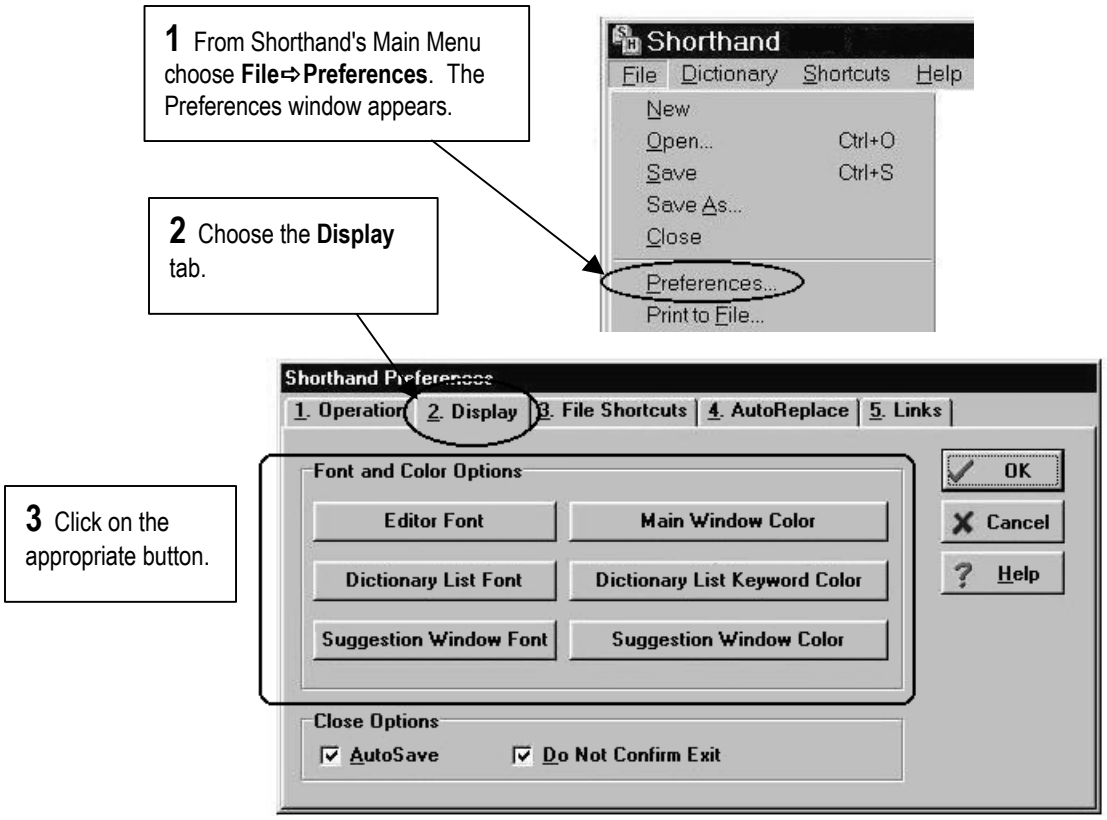
☞ If you are not hearing any beeps after you activate the options for audible warnings, the speaker volume may be at minimum or your speakers may be on MUTE. You can change volume through the Volume icon on the Windows system tray. To change the tone of the audible warnings, go to Windows Control Panel and choose the Sounds icon.

Fonts and colors

To improve readability, you can change the window colors and fonts of some of the text displayed by Shorthand. You can change the following:

- The text fonts displayed in the text input boxes.
- The text fonts displayed in the Dictionary list in the Main Window.
- The text fonts displayed in the Suggestion Window.
- The color of the Main Window.
- The color of the keywords in the Dictionary list in the Main Window.
- The color of the Suggestion Window.

To access the font and color options:



Notes

- Selecting a fixed spaced font such as **`Courier`** makes it easier to spot unwanted spaces in your text.
- The above options only affect how text appears inside Shorthand and do not affect how Shorthand transfers text. See p. 70 to control how the text appears in your word processor.

Automatic File Saving

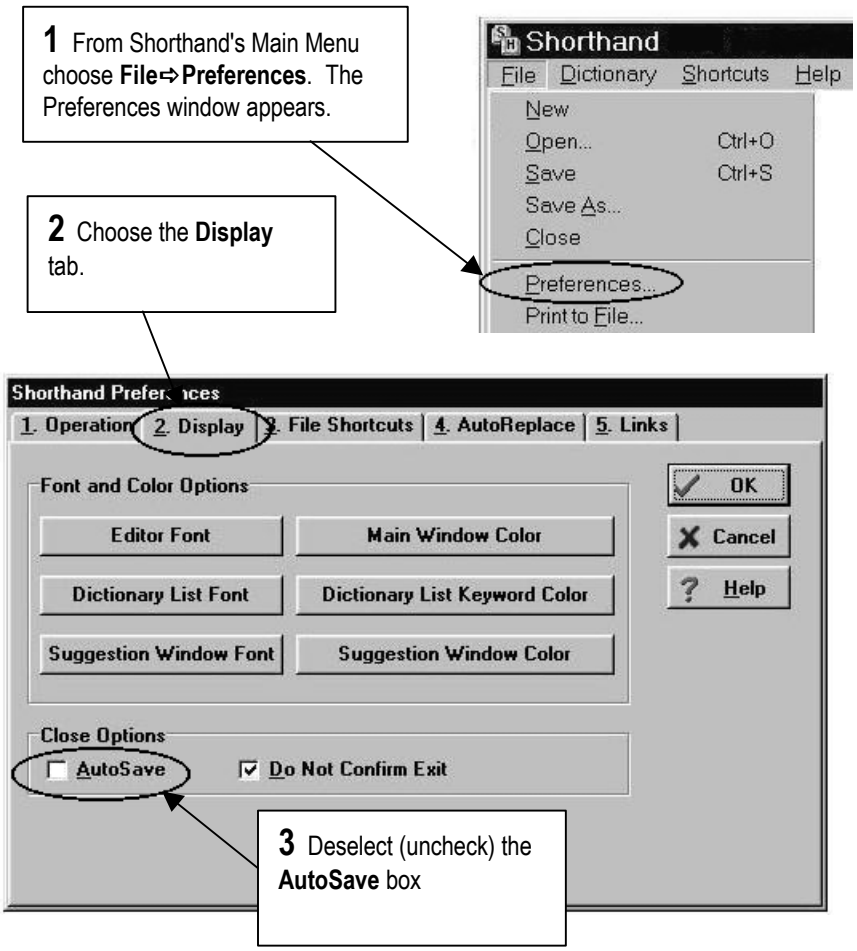
Shorthand automatically saves the active dictionary whenever you switch to another dictionary. In addition, if *automatic file saving* is enabled (the default), Shorthand will also save your file whenever you do any of the following:

- Click on the **Hide** button.
- Exit Shorthand

We suggest you keep automatic saving enabled as this feature may prevent data loss if your computer crashes or you accidentally turn off your computer.

However, if you have a very large dictionary with tens of thousands of entries, it may take a long time (several minutes) to save the file every time you click the Hide button so you may want to disable automatic file saving.

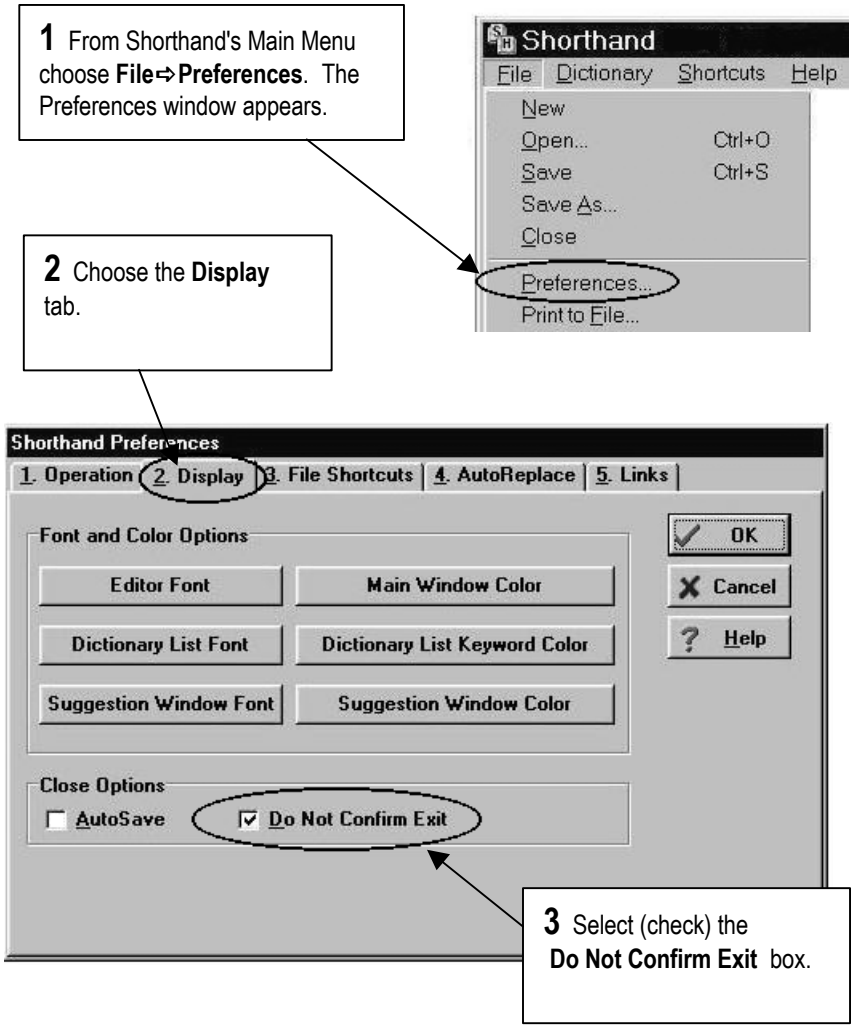
To disable automatic saving:



How to exit Shorthand without confirmation

When you exit (terminate) the Shorthand application by choosing **File⇨Exit** Shorthand will display a dialog box asking if you really want to exit; this helps you avoid shutting down Shorthand accidentally.

If you don't want Shorthand to show the exit confirmation dialog box:



How to use Shorthand on a Network

Warning! Before installing Shorthand on a network, make sure you have a valid Shorthand Site License to cover all persons *able* to use or access Shorthand on your network; unauthorized use of Shorthand by unlicensed persons is a violation of copyright laws and may result in severe civil and criminal penalties.

Even though it is possible to install a single, shared copy of the Shorthand program files on your network and keep each user's configuration files separate through Shorthand's **/cfg** command line parameter (see Shorthand's on-line help: choose **Help⇨Contents** , click on *Advanced Topics* then *Using Shorthand on a Network*), we present below a simpler solution.

Install one copy of Shorthand for each user

The easiest way to use Shorthand on a network is to simply install one copy of Shorthand per user. The Shorthand program files occupy very little disk space (less than 4Mb). To prevent one user from accessing another's Shorthand files, each copy of Shorthand should be installed to a user's private folder on the network.

Specify shared dictionaries as linked dictionaries.

Dictionaries that are to be shared among all users should be placed in a public folder on your network. Shared dictionaries should be loaded into Shorthand as *linked dictionaries* (p. 40). In addition, to prevent accidental modification of shared dictionaries, the shared dictionary files should have a READ ONLY attribute (you can use Windows Explorer to change a file's attributes).

Create a private dictionary for each user.

In each user's Shorthand program folder, create (p. 18) a dictionary to hold the user's private word list. For illustration purposes, we will call this dictionary PERSONAL.SPF. When a user starts Shorthand, Shorthand will open the PERSONAL.SPF as the active (i.e. main) dictionary and open the shared dictionaries as linked files. If a user adds a new entry (p. 24), the new entry is written to PERSONAL.SPF. If a user wishes to override (replace) an entry in a shared dictionary, he/she should:

1. Unload the shared dictionaries by disabling links (see *How to enable/disable links* on p. 41),
2. Create the replacement entry (see *How to add new entries to a dictionary* on p. 24), and
3. Enable links (p.41) to reload the shared dictionaries. The replacement entry will be saved in PERSONAL.SPF and override entries with the same keyword in shared dictionaries.

Chapter 4 Shorthand Tags

Overview

Tags are special fields that can be inserted in your Shorthand text (i.e. the text you place in the Text to Type box). Tags represent special functions in Shorthand and allow you to do things such as:

- Insert the current date or time.
- Display a dialog box to the user and insert the user's response into the Shorthand text.
- Import the contents of a text file.
- Pause between characters.
- Simulate keystroke combinations such as **Shift+F1**.
- Execute Tcl (*Tool Command Language)* scripts and insert the result into your Shorthand text.

The basic syntax of a Tag is:

{@tag_name *arg1 arg2 arg3* ...}

where *tag_name* is the name of the Tag (e.g. PAUSE to create a short pause) and *arg1*, *arg2*, *arg3* are arguments and options used by the Tag (e.g. number of seconds to pause for the PAUSE tag).

When are Tags evaluated?

When you tell Shorthand to expand a keyword in your word processor, Shorthand first evaluates all tags that begin with {@INPUT} (such as {@INPUT}, {@INPUTTCL} and {@INPUTFILE}). After all @INPUT tags are evaluated, Shorthand simulates backspaces to remove the keyword in your word processor and proceeds to insert text to your word processor; the non-input tags (e.g. {@KEY}, {@PAUSE}, {@SHORTDATE}) are evaluated at this time.*

Date/time tags

Shorthand has several tags that will insert the date/time in your Shorthand text.

{@INPUTDATE}	Displays a dialog box in which the user can enter a date. The output can be in short or long date format.
{@LONGDATE}	Inserts current date using the Windows long date format. Example output: Saturday, July 14, 2001
{@SHORTDATE}	Inserts current date using the Windows short date format. Example output: 07/14/01
{@LONGTIME}	Inserts current time using the format hours:minutes:seconds. Example output: 11:43:05 AM
{@SHORTTIME}	Inserts current time using the format hours:minutes. Example output: 11:43 AM

Keystroke tags

Shorthand provides several tags for simulating special keystrokes (e.g. **F1**) and keyboard combinations (e.g. **Shift+Ctrl+F1**).

{@KEY *key*}	Simulates a complete keystroke. Example: {@KEY shift+f10} simulates **Shift+F10**
{@KEYDOWN *key*}	Simulates the pressing (holding down) of a key. Example: This tag simulates holding down the shift key: {@KEYDOWN shift}
{@KEYUP *key*}	Simulates the release of a key. Example: This tag simulates releasing the shift key: {@KEYUP shift}
{@PAUSE}	Inserts a 1 second pause before inserting the next character.

File tags

Shorthand has a tag to import the contents of a text file into your Shorthand text.

{@INPUTFILE *filename*}	Inserts the contents of filename. The file may contain Shorthand tags.

Input tags

Shorthand provides several tags for displaying dialog boxes to the user; the user can type text into the dialog box which will be inserted in your Shorthand text. With input tags, you can use Shorthand to show a series of "question and answer" dialog boxes to create custom reports.

{@INPUT *prompt*}	Displays a dialog window for entering text. The *prompt* is displayed at top of the window.
{@INPUT *list_items*}	Displays a dialog window showing a selectable list of items.
{@INPUTDATE}	Displays a dialog window for selecting a calendar date.
{@INPUTMSG *message*}	Displays a message.

Miscellaneous tags

{@NOSPACE}	Prevents a space from being added after a keyword is expanded.
{@DELETELINE}	Deletes an entire line of text.
{@REM *comment*}	Adds comments (remarks). This tag will not be typed out.
{@LEFTBRACE}	Inserts a left brace **{**
{@RIGHTBRACE}	Inserts a right brace **}**
{@INPUTTCL}	Executes a Tool Command Language (Tcl) script.

How to simulate keyboard commands

You can use Shorthand's {@KEY} tag to simulate keyboard commands recognized by your word processor. It is important to remember that Shorthand can only simulate keystrokes and has no notion of what the commands do in your word processor. For example, you can use Shorthand to simulate the keyboard command turn bolding on/off but the text in Text to Type box will not be displayed in bold.

The syntax of the {@KEY} tag is:

{@KEY *key n*}

where *key* is name of the keystroke you wish to simulate and *n* is the number of times to repeat the keystroke. For example, {@KEY f} will simulate pressing and releasing the E key and {@KEY home 3} will simulate the HOME key 3 times.

Here are some examples of {@KEY} commands:

1	`{@KEY 1}`
!	`{@KEY !}` or `{@KEY shift+1}` (assumes that ! is on the 1 key.)
F10	`{@KEY f10}`
ALT + F7	`{@KEY alt+f7}`
CTRL + E	`{@KEY ctrl+e}`
SHIFT + A	`{@KEY shift+a}`
CTRL + ALT + E	`{@KEY ctrl+alt+e}`
DEL	`{@KEY del}`
END	`{@KEY end}`
PG UP	`{@KEY pgup}`
PG DN	`{@KEY pgdn}`
ENTER	`{@KEY enter}`
SPACE	`{@KEY space}`
↹ (TAB)	`{@KEY tab}`
ESC	`{@KEY esc}`
↖ (BACKSPACE)	`{@KEY bksp}`

↑	`{@KEY up}`
↓	`{@KEY down}`
←	`{@KEY left}`
→	`{@KEY right}`
+ on main keyboard	`{@KEY +}`
+ on numeric keypad	`{@KEY NumPad+}`
/ on main keyboard	`{@KEY /}`
/ on numeric keypad	`{@KEY NumPad/}`
{ (left brace)	`{@LEFTBRACE}`
} (right brace)	`{@RIGHTBRACE}`

Examples

In most Windows applications, pressing **Ctrl+Shift+Left Arrow** selects the previous word so the following Shorthand code can be used to delete the previous word:

```
{@KEY Ctrl+Shift+Left}{@KEY del}
```

Most Windows applications also have a Save command under a File menu which you can access by first pressing **Alt+F** followed by pressing **S**. So the Shorthand code to save a file would be:

```
{@KEY Alt+F}s
```

The following pages contain more examples on how to use the {@KEY} tag.

Knowing your word processor s keyboard shortcuts will help you use Shorthand more effectively.

To find the keyboard commands for your word processor, bring up your word processor's on-line help and search for "shortcuts". Word processors such as Microsoft Word and WordPerfect allow you to define keyboard shortcuts to apply different font styles, create tables, insert locked spaces, etc. Once you know the keyboard command for a particular function (e.g. enable/disable bold face) you can use the {@KEY} tag to simulate that command with Shorthand (see p. 70 for an example).

How to use the {@KEY} tag to change font styles

You can use Shorthand's {@KEY} tag to simulate keyboard commands recognized by your word processor to change how the text inserted by Shorthand appears in your word processor.

Example problem

How do I use Shorthand to simulate superscripts to type out the symbol for degrees Fahrenheit, **^{0}F,** in Microsoft Word?

Solution

Create a new Shorthand entry (p. 24) with the following data:

Keyword: `df`

Text to Type:

```
{@KEY ctrl+shift++}0{@KEY ctrl+shift++}F
```

Test the entry

1) Make sure Shorthand is minimized. If any of Shorthand's dialog boxes is visible close it and click on the **Hide** button in the Shorthand Main Window.

2) Bring up Microsoft Word and type:

`98` SPACE `df` SPACE `.`

The result should be: 98 ^{0}F

How it works

Here's an explanation of the Shorthand code in the Text to Type box:

`{@KEY ctrl+shift++}`	Simulates CTRL + SHIFT + + to turn on the superscript font style in Microsoft Word.
`0`	Types out "0" (which will appear in superscript font).
`{@KEY ctrl+shift++}`	Simulates CTRL + SHIFT + + to turn off superscript.
`F`	Types out "F" in normal font.

You can use this same technique to simulate other fonts; for example, {@KEY Ctrl+B} will enable/disable bold fonts in Microsoft Word. See your word processor's documentation for the keyboard commands to change fonts.

How to use the {@KEY} tag to insert special codes such as locked spaces

You can use Shorthand's {@KEY} tag to simulate keyboard commands recognized by your word processor to insert special symbols or characters.

Example problem

In Microsoft Word, how do I use Shorthand to create a entry for "mm" (millimeter) with a locked space before "mm"? A *locked space* (also known as a *non-breaking space*) tells the word processor to keep two words on the same line.

Solution

Create a new Shorthand entry (p. 24) with the following data:

Keyword: `mm`

Text to Type:

```
{@KEY bksp}{@KEY ctrl+shift+space}mm
```

Test the entry

1) Make sure Shorthand is running and minimized (click Shorthand's Hide button).
2) Bring up Microsoft Word.
3) In your word processor, type: `5` SPACE `mm` SPACE .

The result should be: `5 mm`

with a locked space between `5` and `mm` .

How it works

The Microsoft Word command to insert a locked space is **Ctrl+Shift+Space** (i.e. while holding down the CTRL and SHIFT keys, press the SPACE BAR). Here's an explanation of the Shorthand code in the Text to Type box:

`{@KEY bksp}`	Simulate a BACKSPACE to delete the previous character. This is necessary to prevent double spaces.
`{@KEY ctrl+shift+space}`	Simulates **Ctrl+Shift+Space** to insert a locked space in Microsoft Word.
`mm`	Types out "mm".

How to pause Shorthand in the middle of an expansion

You can use the {@PAUSE} tag to insert a delay/time lapse between keystrokes or after a keyboard command to allow the computer to process the command. Complicated commands (especially those that launches applications or displays new windows) may not be properly processed if the computer is deluged with dozens of keyboard commands occurring in a few seconds.

The syntax for the {@PAUSE} tag is:

{@PAUSE *n*}

where *n* is the number of seconds to pause. For example, {@PAUSE 0.5} will tell Shorthand to pause half a second. If you don't specify *n*, Shorthand will pause 1 second. If *n* is larger than 10, Shorthand will pause only 10 seconds at most.

Example problem

How do I create a keyboard shortcut to move the cursor to the next occurrence of the word "INPUT:" in a Microsoft Word document?

Solution

Create a new Shorthand entry (p. 24) with the following data:

Keyword: `find_input` Shortcut: `Ctrl+Alt+I`

Text to Type:

```
{@KEY alt+e}{@KEY f}{@PAUSE}INPUT:{@PAUSE}{@KEY alt+f}
```

Test the entry

1) Make sure Shorthand is running and minimized (click Shorthand's Hide button).

2) Bring up Microsoft Word and create or open a document that contains several instances of the word "INPUT:".

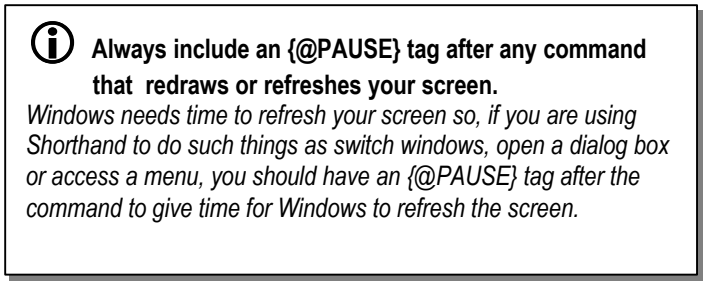
3) In Microsoft Word, press: **Ctrl+Alt+I**

You should see Word bring up the Find dialog box, search for the next instance of "INPUT:" then close the Find dialog box.

How it works

Shorthand uses Word's Find function to search for INPUT; here's an explanation of the Shorthand text in the Text to Type box:

`{@KEY alt+e}`	Simulate **Alt+E** to choose **Edit** from Word's main menu.
`{@KEY f}`	Chooses **Find** from Word's Edit menu.
`{@PAUSE}`	Pauses 1 second to give Word time to display the **Find and Replace** dialog box.
`INPUT:`	Types out "INPUT:" in the **Find what** box.
`{@PAUSE}`	Pauses 1 second to give Word time to update the **Find what** box.
`{@KEY alt+f}`	Chooses the **Find Next** button to start the search.



Always include an {@PAUSE} tag after any command that redraws or refreshes your screen.

Windows needs time to refresh your screen so, if you are using Shorthand to do such things as switch windows, open a dialog box or access a menu, you should have an {@PAUSE} tag after the command to give time for Windows to refresh the screen.

How to insert the current date and time

Example problem

How do I create a Shorthand entry to type out the following sentence:

The patient was admitted on (today's date) at (current time).

Solution

We use the {@LONGDATE} and {@SHORTTIME} tags to display the current date and time. The techniques shown in this section apply as well to the other date/time tags (p. 66).

Create a new Shorthand entry (p. 24) with the following data:

Keyword: `time2`

Text to Type:

```
The patient was admitted on {@LONGDATE} at {@SHORTTIME}.
```

Shorthand uses the Windows Date settings to display the style of the date string. You can change the Date style as follows: go to the Windows Control Panel, choose Regional Settings, choose Date. You will need to restart Shorthand to apply the new style.

Test the entry

1) Make sure Shorthand is minimized. If any of Shorthand's dialog boxes is visible close it and click on the Hide button on the Shorthand Main Window.

2) Bring up your favorite word processor.

3) In your word processor, type: `time2` followed by SPACE.

`time2` should be replaced with something like:

The patient was admitted on Saturday, February 10. 2001 at 3:45 PM.

How it works

Here's an explanation of the Shorthand code in Text to Type box:

`The patient was admitted on`	Types out "The patient was admitted on"
`{@LONGDATE}`	Types out the current date using the Windows Long format.
`at`	Types out "at".
`{@SHORTTIME}`	Types out the current time in hour:minute AM/PM format.
`.`	Types out a period.

How to ask for a date from the user

You can use Shorthand's {@INPUTDATE} tag to display a dialog box to retrieve a date response from the user.

The syntax of the {@INPUTDATE} tag is:

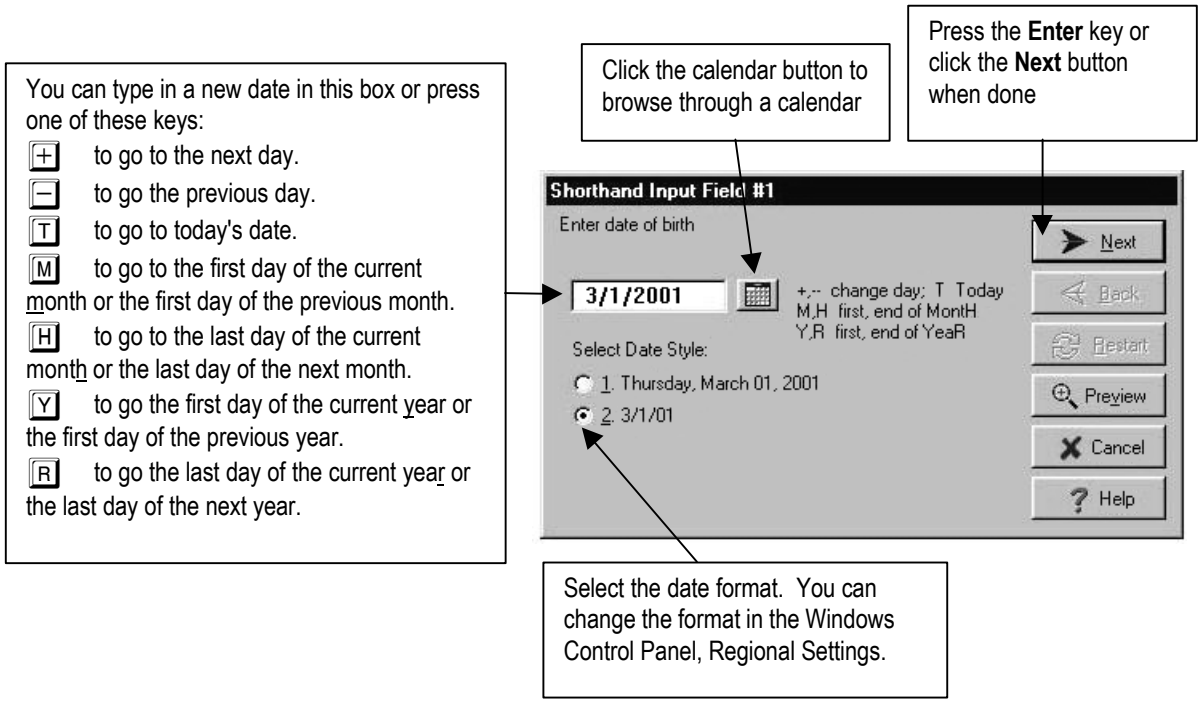
{@INPUTDATE *prompt* @LONG }

where *prompt* is the question to be displayed at the top of the Input dialog box. "@LONG" is an optional argument. If "@LONG" is present, the Windows Long Date format will be made the default selection; if @LONG is not specified, the Windows Short Date format is used.

How to use {@INPUTDATE}

The {@INPUTDATE} tag works just like the {@INPUT} tag (p. 76) except that {@INPUTDATE} will only accept a valid date.

When Shorthand encounters an {@INPUTDATE} tag, Shorthand will display a dialog box to get a date from the user:



How to get input from the user

You can use Shorthand's {@INPUT} tag to display a dialog box with a question and retrieve the user's answer. By using a series of {@INPUT} tags you can create a "questionnaire" to guide the user in entering needed information.

> **When are {@INPUT} tags processed?**
> *Shorthand evaluates all {@INPUT} tags before inserting any text into your word processor. This means that if the user types in an {@INPUT} tag (or any other Shorthand tag), these tags will be recognized and processed by Shorthand. But this also means that you cannot use an @INPUT tag to change Shorthand's behavior once expansion has already started.*

The syntax of the {@INPUT} tag is:

{@INPUT *prompt* @WANTRETURNS}

where *prompt* is the question to be displayed at the top of the Input dialog box.

@WANTRETURNS is an optional argument and controls what happens when you press the ENTER key in the input window. If @WANTRETURNS is present, pressing ENTER will create a new line. If @WANTRETURNS is not present, pressing ENTER is equivalent to clicking on the NEXT button.

Example problem

How do I use Shorthand to type out the following boilerplate text?

Name: (customer_name)

Address: (customer_address)

Telephone: (customer_phone)

Solution

Create a new Shorthand entry (p. 24) with the following data:

Keyword: `cust1`

Text to Type:

```
Name: {@INPUT Customer's last name, first name, middle
initial}  ENTER

Address: {@INPUT Number, street name, city, state, zip
code, country}  ENTER

Telephone: ({@INPUT enter area code})-{@INPUT enter
telephone number}
```

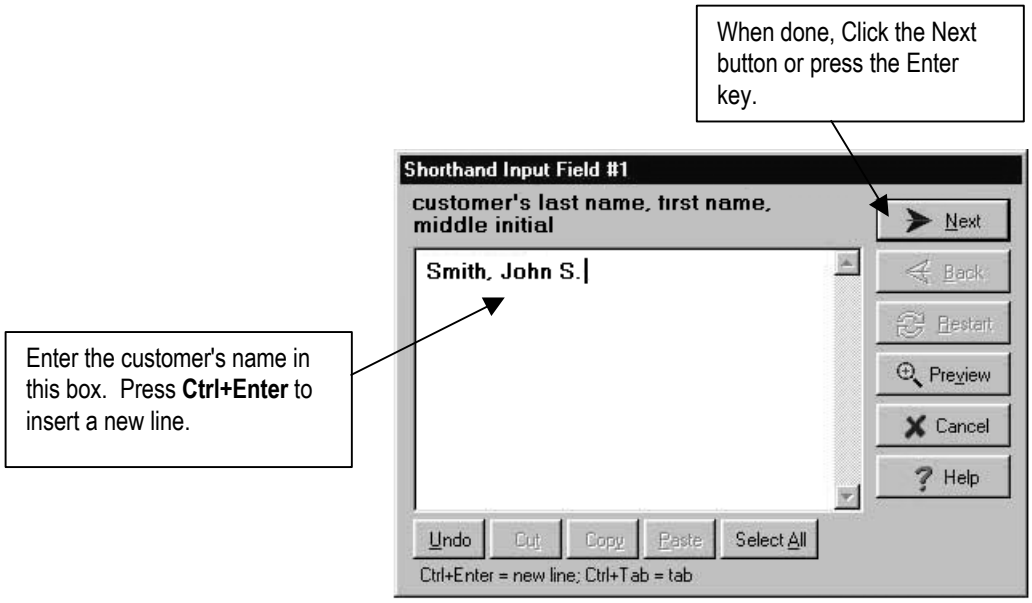
Test the entry

1) Make sure Shorthand is running and minimized (click Shorthand's Hide button).

2) Bring up your word processor and create or open a new document.

3) In your word processor, type: `cust1` ENTER

4) Shorthand displays a dialog box asking you for the customer's name. Enter the customer's name and click on the **Next** button or press ENTER .

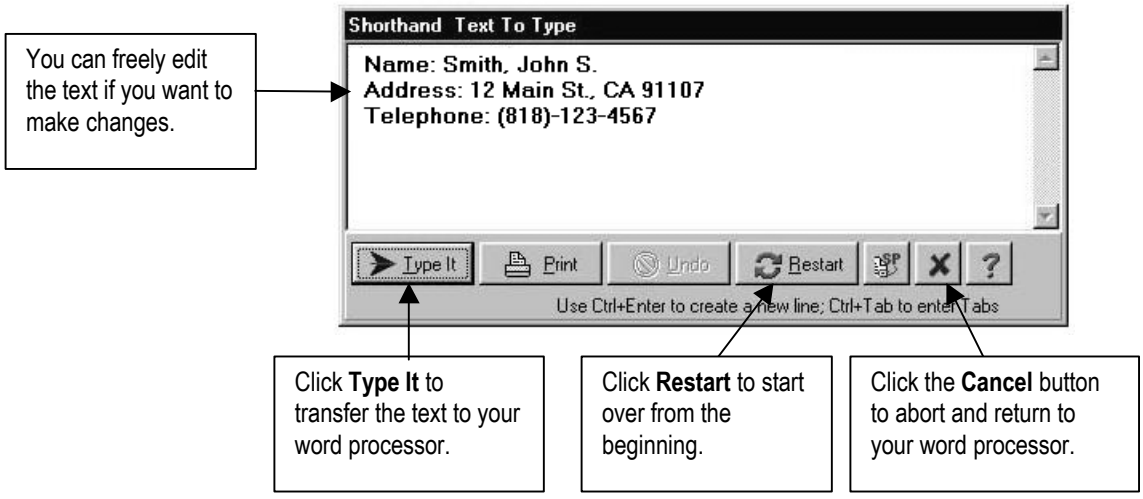


5) Shorthand displays a dialog box asking you for the address. Enter the address on one line. If you wish to enter multiple lines, you can press **Ctrl+Enter** to create a new line. When done, click on the **Next** button or press ENTER .

6) Shorthand displays a dialog box asking for the area code. Enter the area code and click **Next** button or press ENTER .

7) Shorthand displays a dialog box asking for the phone number. Enter the phone number and click the **Next** button.

8) Shorthand displays the Text to Type dialog box showing a preview of the text that will be inserted into your word processor.



How to display user-selectable pick lists

Pick lists are an extended feature of {@INPUT} tags wherein a list of selectable items will appear in a dialog box; the user's selection(s) will then be reported back to Shorthand.

The syntax for the {@INPUT} tag for pick lists is:

{@INPUT ***prompt options*** **@SEP=*"sep"* @SINGLE**
item1
item2
:
itemN
}

where:

- ***prompt*** is the text displayed at the top of the dialog box
- ***options*** are optional arguments and can be one or more of the following:

  @SEP=*"sep* specifies the separator string. If the user selects more than one item, the items will be separated by *sep* (note that *sep* can contain spaces and must be enclosed in double quotes ""). If @SEP is not specified, Shorthand uses ", " as the default separator.

  @SINGLE specifies that only one item can be selected. If @SINGLE is not present, the user can select one or more items.

  @GROUP will make items that begin with an asterisk "*" non-selectable. This is useful for adding subheadings or item breaks (i.e. blank lines between items).

  @GROUPBOLD is the same as @GROUP except that the subheadings will be displayed in bold.

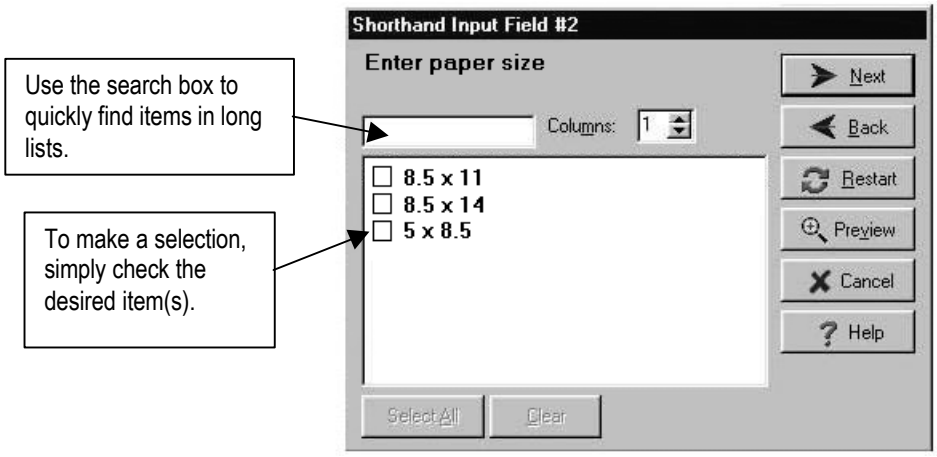
  @GROUPPRINT is the same as @GROUP except that the subheadings will be inserted in the output if at least one of its sub-items is selected.

  @AUTONUMBER adds a number to each item so you can easily jump to an item by typing a number.

- ***input1, input2 ... inputN*** are the items in the pick list. Note that the *inputN* text must be placed on separate lines. For example, the following Text to Type:

```
{@INPUT Enter paper size[ENTER]
8.5 x 11[ENTER]
8.5 x 14[ENTER]
5 x 8.5[ENTER]
}
```

will display the following dialog box:

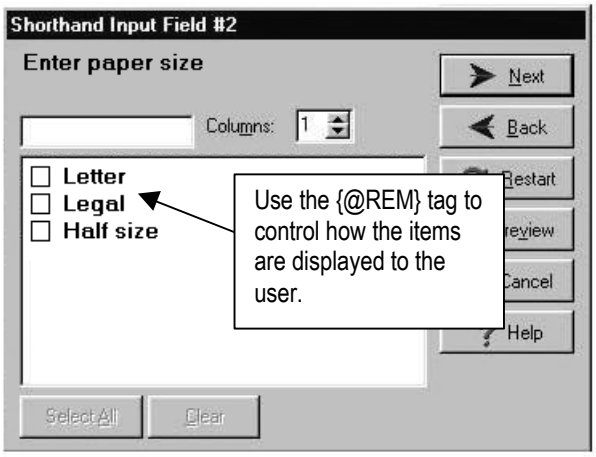


The selected items will then be inserted into your Shorthand text. For example, if the user selects **8.5 x 11**, the string "8.5 x 11" will be inserted in the text Shorthand transfers to your word processor.

You can use more descriptive text to display to user by using the **{@REM *text*}** tag. If an **{@REM *text*}**, tag appears in an {@INPUT} item line, Shorthand will display text in the selection list. For example, the following Text to Type:

```
{@INPUT Enter paper size[ENTER]
{@REM Letter}8.5 x 11[ENTER]
{@REM Legal}8.5 x 14[ENTER]
{@REM Half size}5 x 8.5[ENTER]
}
```

will display a dialog box with the following list of items:



To make a selection, the user simply checks the desired item.

The selected item will then be inserted into your Shorthand text. For example, if the user selected **Letter**, the string "8.5 x 11" will be inserted in your Shorthand text.

Example problem

How do I use Shorthand to type out the following sentence?

Upon entry, the patient was suffering from: (list of symptoms).

where the list of symptoms can be one or more of:

Dizziness, Headache, Abdominal Pain, Fever

If the patient has more than one symptom, the symptoms are to be separated by slashes "/".

Solution

Create a new Shorthand entry with the following data:

Keyword: `symps1`

Text to Type:

```
Upon entry, the patient was suffering from:  {@INPUT Choose the
symptoms that apply @SEP="/"[ENTER]
Dizziness[ENTER]
Headache[ENTER]
Abdominal Pain[ENTER]
Fever[ENTER]
}
```

Test the entry

1) Make sure Shorthand is running and minimized (click Shorthand's Hide button).

2) Bring up your word processor and create or open a document.

3) In your word processor, type: `symps1` [SPACE]

4) Shorthand displays a dialog box asking you for the symptoms with a list of choices. Select one or more items and click on the **Next** button or press [ENTER] .

5) Shorthand displays the Text to Type dialog box showing a preview of the text that will be inserted into your word processor. If you see a mistake, you can make changes in this window or click the **Restart** button to start over. Click the **Type It** button or press [ENTER] to insert the text to your word processor.

How it works

Here's an explanation of the code in the Text to Type box:

`Upon entry, the patient was suffering from:`	Types the text as is.
`{@INPUT Choose the symptoms that apply`	Displays an input dialog with the prompt "Choose the symptoms that apply".
`@SEP="/"`	Tells Shorthand to place a "/" between selected items.

`Dizziness`
`Headache`
`Abdominal Pain`
`Fever`
	These are the items that will be appear in the selection list.

`}`	Closing brace for the {@INPUT} tag.

☞ The separator string specified by @SEP can be any string including {@KEY} codes. For example, to place each selected item in a new line, specify:

```
@SEP="{@KEY enter}"
```

ⓘ *You can navigate the pick list window entirely through the keyboard: Use the arrow keys on your keyboard to scroll up and down the list and press SPACE to select/deselect an entry. Press ENTER to go to the next screen.*

ⓘ **Tags can be embedded in pick lists.**
Pick lists can contain any valid Shorthand text, including tags. The only limitation is that each pick list entry has to be a single line. For example, if you wish to have a pick list call up another pick list, you can save the second pick list in a text file and then embed an {@INPUTFILE} tag (see p. 82 for how to insert files in Shorthand text) in the first pick list. We work around the one-line per entry limitation of pick lists by saving the second pick list in a file and using the {@INPUTFILE} tag to read the file.

How to insert a text file

You can insert the contents of a text file into your Shorthand text with the {@INPUTFILE} tag. The syntax of the tag is:

{@INPUTFILE *filename*}

where *filename* is the name of the file to import. The {@INPUTFILE} tag can read only plain text files (i.e. files that you can read with NOTEPAD); files such as Word .doc files and .JPG files cannot be imported into Shorthand (but see *How to insert non-text files into your word processor* on p. 84). If the text file imported by {@INPUTFILE} contains Shorthand tags, the tags will be evaluated by Shorthand.

Example problem

I have several Shorthand entries that contain my phone number. How do I set up the entries so that I don't have to modify each entry if my phone number changes?

Solution

1) Create a text file called **phone.txt** (using Windows NOTEPAD) containing your phone number:

```
(999) 123-4560
```

2) Save the phone.txt in your **C:\My Documents** folder.

3) Modify your Shorthand entries that contain your phone number to import in the text file. For example:

Keyword: `q1`

Text to Type:

```
If you have questions, please call me at {@INPUTFILE  "c:\my
documents\phone.txt"}.
```

4) If the phone number changes, we only need to modify the **phone.txt** file.

Test the entry

1) Make sure Shorthand is running and minimized (click Shorthand's Hide button).
2) Bring up your favorite word processor.
3) In your word processor, type:

   `q1` SPACE

The result should be:

If you have questions, please call me at (999) 123-4560.

How it works

Here's an explanation of the Shorthand code in the Text to Type box:

`If you have questions, please call me at`	Types out "If you have questions, please call me at".
`{@INPUTFILE "c:\my documents\phone.txt" }`	Reads in the contents of phone.txt. Note that since the pathname is enclosed in double quotes because the name contains a space. If we left out the double quotes, Shorthand will attempt to read in the file called **c:\my** which is not what we want.
`.`	Types out a period.

You can use the {@INPUTFILE} tag to get around the 32,000 character limit in the Text to Type box.

The Text to Type box can support a maximum of 32,000 characters. You can increase this limit to 64,000 characters by saving your text in a file then using the {@INPUTFILE} tag to read in the file. Shorthand has an internal limit of 64,000 characters for long forms; for files larger than 64,000 characters, you can use Shorthand s {@KEY} tag to simulate your word processor s commands to insert the file directly into your word processor (see p. 84).

How to insert non-text files into your word processor

You can use Shorthand's {@KEY} tag to simulate the keyboard commands to import a non-text file into your word processor.

Example problem

My company's logo is saved in a file called "logo.jpg" in the C:\My Documents folder. How do I create a Shorthand entry to insert the logo into a Microsoft Word document?

Solution

Create a new Shorthand entry with the following data:

Keyword: `logo1`

Text to Type:

```
{@KEY alt+i}pf{@PAUSE}c:\My Documents\logo.jpg{@PAUSE}{@KEY alt+r}
```

Test the entry

1) Make sure Shorthand is running and minimized (click Shorthand's Hide button).
2) Bring up Microsoft Word and create or open a document.
3) Type: `logo1` SPACE

Microsoft Word will insert logo.jpg into the document.

How it works

Here's an explanation of the Shorthand code in the Text to Type box:

`{@KEY alt+i}`	Simulates **Alt+I** to bring up Word's **Insert** menu.
`p`	Chooses **Picture** from Word's **Insert** menu..
`f`	Chooses **From File** from the **Picture** submenu.
`{@PAUSE}`	Pauses 1 second to allow time for Word to display the **Insert Picture** dialog box.
`c:\my documents\logo.jpg`	Types the filename in the **File name** box.
`{@PAUSE}`	Pauses a second to give Word time to accept the filename.
`{@KEY alt+r}`	Simulates **Alt+R** to choose the **Insert** button.

How to prevent extra spaces after an expansion

When you type SPACE (or any non-alphanumeric key) after a keyword, Shorthand will expand the keyword *and* insert the SPACE (or non-alphanumeric character) you typed. There are times when you just want to expand a keyword and don't want the SPACE to be inserted; to prevent the extra space, specify the {@NOSPACE} tag anywhere in your Text to Type.

The syntax for the {@NOSPACE} tag is:

{@NOSPACE}

and it can be placed anywhere in your Shorthand text.

Example problem

I have a keyword called **subj** which will type out a heading called "SUBJECTIVE:" followed by exactly two new lines. For some reason, I'm getting three new lines when I type `subj` ENTER . Here is my definition:

Keyword: `subj`

Text to Type:

```
SUBJECTIVE:{@KEY Enter}{@KEY Enter}
```

Solution

The extra line is coming from the ENTER key you typed after the keyword. To prevent the extra line, add {@NOSPACE} in your Shorthand text:

Keyword: `subj`

Text to Type:

```
SUBJECTIVE:{@KEY Enter}{@KEY Enter}{@NOSPACE}
```

The TAB key expands keywords without inserting a space.

*An alternative way to prevent extra spaces is to press the **TAB** key after typing the keyword. Pressing TAB after a keyword will immediately expand the keyword without inserting extra spaces.*

How to remove unwanted lines

The {@DELETELINE} tag removes the entire line it is on (we define a "line" here to mean a series to characters that ends with a [ENTER] in the Text to Type box).

The syntax for the {@DELETELINE} tag is:

{@DELETELINE}

Example problem

I have a series of {@INPUT} pick list tags to get input from the user but some of the inputs should not be typed out if the question is not applicable. How can I get rid of unneeded @INPUT tags?

Solution

The solution is to add an item to your pick list to delete the @INPUT tag. For example, let's expand the example on p. 80 to add a "not applicable" option:

Text to Type:

```
Upon entry, the patient was suffering from:  {@INPUT Choose the
symptoms that apply @SEP="/"[ENTER]
Dizziness[ENTER]
Headache[ENTER]
Abdominal Pain[ENTER]
Fever[ENTER]
{@REM Not Applicable}{@DELETELINE}[ENTER]
}
```

Now when the user selects "Not Applicable", Shorthand will process the {@DELETELINE} tag and the line "Upon entry, the patient was suffering from..." will not be typed out to your word processor.

How to use variables

The results of {@INPUT} tags can be stored into a *variable*. By placing the variable in your Text to Type, you can insert the result of an {@INPUT} tag without having to ask the same question again.

To tell Shorthand to save the result of an {@INPUT} tag to a variable, you enter a variable name enclosed by parentheses directly after the tag name. For example:

{@INPUT(*var*) *prompt*}

{@INPUTDATE(*var*) *prompt*}

{@INPUTFILE(*var*) *prompt*}

where *var* is the name of the variable. The variable name can be any alphanumeric string without any spaces.

When Shorthand processes an {@INPUT} tag with a variable, Shorthand will replace all instances of the string **{*var*}** with the result of the {@INPUT} tag (where ***var***, again, refers to the name of the variable).

Example problem

How do I create a Shorthand entry for the following paragraph with STATE replaced by the state specified by the user?

> Choice of Law. This agreement will be governed by the laws of STATE and you agree that any claims regarding the Product shall be brought in STATE, and waive any objections to jurisdiction in the U.S. District Court for the District of STATE or the STATE Superior Court.

Solution

Here's how the Shorthand code should look like:

Text to Type:

```
Choice of Law. This agreement will be governed by the laws of
{@INPUT(STATE) Enter state} and you agree that any claims regarding
the Product shall be brought in {STATE}, and waive any objections to
jurisdiction in the U.S. District Court for the District of {STATE}
or the {STATE} Superior Court.
```

How it works

We use an {@INPUT} tag to get the name of the state from the user and store the result to a variable named STATE. Shorthand will then replace all instances of {STATE} with the string entered by the user in the Input dialog box.

How to output braces

The {@LEFTBRACE} tag outputs a left brace: **{**.

The {@RIGHTBRACE} tag outputs a right brace: **}**.

The {@LEFTBRACE} and {@RIGHTBRACE} tags are useful if you want Shorthand to type out strings that can be interpreted as tags.

Example problem

How do I create a Shorthand entry for the following paragraph with the string "{@INPUT}" actually typed into my word processor?

> The Shorthand tag to display an input window is: {@INPUT}.

Solution

Here's how the Shorthand code should look like:

Text to Type:

```
The Shorthand tag to display an input windows is:
{@LEFTBRACE}@INPUT{@RIGHTBRACE}
```

How it works

We use the {@LEFTBRACE} and {@RIGHTBRACE} tags to display the { and } characters.

Here's an explanation of the code in the Text to Type box:

`The Shorthand tag to display an input windows is:`	Shorthand types the text as is.
`{@LEFTBRACE}`	Shorthand types a **{**
`@INPUT`	Shorthand types the text as is.
`{@RIGHTBRACE}`	Shorthand types a **}**

Chapter 5 Tcl Scripts

Overview

The {@INPUTTCL} tag is a powerful programming feature that allows you to embed programming code in your Shorthand text.

Tcl (pronounced “tickle”) stands for Tool Command Language and is a popular scripting language for “gluing” applications and text data. Tcl was developed by John Ousterhout at the University of California; the Tcl libraries, source code, as well as many tools and extensions are available free from the Internet.

A Quick Introduction to using Tcl

Tcl code looks very much like the commands you enter in the Windows console (or DOS window).

For example, the Tcl code

```
cd /tmp/dir1
```

will change the current working directory to /tmp/dir1.

The Tcl code

```
file rename john.txt fred.txt
```

will rename a file called “john.txt” to “fred.txt”.

To give you an idea of the power of Tcl, below is a brief sampling of the standard commands available in Tcl.

String Commands

`format`	Creates formatted output.
`regexp`	Matches strings using powerful regular expressions (e.g. *.*).
`regsub`	Performs a search and replace.
`scan`	Parses and extracts values from a string.
`string compare`	Compares 2 strings.
`string length`	Returns the length of a string.
`string range`	Extracts a substring from a string.

List Commands

`lappend`	Appends an element to the end of a list.
`lindex`	Returns a list element.
`linsert`	Inserts an element into a list.
`llength`	Returns the number of elements in a list.
`lreplace`	Replaces a range of list elements.
`lsearch`	Searches a list.
`lsort`	Sorts a list.

Control Flow Commands

`if`	Conditionally evaluates a script.
`foreach`	Loops through a script.
`while`	Loops through a script.
`break`	Exits a loop.
`continue`	Continues execution at the start of a loop.

File Commands

`cd`	Changes the current working directory.
`pwd`	Returns the current directory.
`glob`	Returns a list of files with pattern matching.
`open`	Opens a file for reading/writing.
`close`	Closes an open file.
`seek`	Sets the file s access position.
`read`	Reads from a file.
`puts`	Writes to a file.
`source`	Reads and evaluates a script stored in a file.
`file copy`	Copies a file.
`file delete`	Deletes a file.
`file rename`	Renames a file.
`file exists`	Determines if a file exists.
`file size`	Returns the file size.
`file atime`	Returns the file time.
`file dirname`	Returns the file s directory.
`file extension`	Returns the file s extension.
`file rootname`	Returns the file s root name.
`file isdirectory`	Determines if a string is the name of a file or a directory.

System Commands

`exec`	Executes an external Windows application.
`clock seconds`	Returns current time in seconds.
`clock format`	Returns a formatted date/time string.
`dde`	Communicates with other applications through the Windows Dynamic Data Exchange.

Errors and Exceptions

`catch`	Executes a specified script if an error occurs.
`error`	Generates an exception and aborts the script.

Math Commands

`incr`	Increments (adds 1 to) the value of a variable.
`expr`	Evaluates a mathematical expression, e.g. expr 10.1 * 0.08. In addition to the basic arithmetic operations (+, -, *. /), Tcl also provides advanced math functions for computing square roots, logarithms, sine/cosine, exponentials, etc.

An important feature of Tcl is that you are not limited to the commands listed in the preceding pages. Tcl is *extensible* which means new commands can be added to increase the functionality of Shorthand:

Shorthand Extension Commands

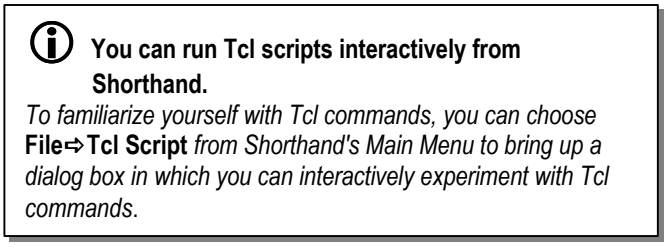
`run`	Launches a Windows file or program (see p. 98).
`sh_file`	Manages Shorthand dictionaries (e.g. open, close or save a dictionary) (see p. 124).
`sh_input`	Displays an input dialog box (see p. 112).
`sh_list`	Manages your word lists (e.g. add an entry, search for a keyword, modify the text to type) (see p. 100).
`sh_set`	Sets the Shorthand parameters (e.g. turn AutoReplace on/off) (see p. 104).

Refer to Shorthand's on-line help for the technical details on the Shorthand Extension commands; examples on how to use some of the extension commands are presented in the following pages.

Tcl Resources

Teaching Tcl programming is beyond the scope of this book. To get started with Tcl, it is recommended you visit the many Tcl web sites on the Internet (e.g. `http://www.scriptics.com`). A very good Tcl book is *Practical Programming in Tcl and Tk (3rd Edition)* by Brent B. Welch (Addison-Wesley, 2000; ISBN: 0130220280).

Shorthand installs only the "core" Tcl scripting language and does not load the *Tk* libraries (Tk is a windowing/graphics library extension for Tcl); when reading a book on Tcl, you can concentrate only on the first few chapters that explain the basic Tcl commands and skip the sections on Tk.



Guidelines for writing Tcl scripts in Shorthand

Keep Tcl scripts short and simple.

Short scripts are easier to debug and understand. If you need to do something complex, consider writing it as a separate application using a high level language such as Visual Basic which can be launched from Shorthand with the Tcl `run` command.

Use the sh_input Tcl command to get data from the user.

You can use the `sh_input` command to display dialog boxes to the user. See Shorthand's on-line help for the syntax of these commands. The following example asks the user for a name and returns the name in uppercase:

```
{@INPUTTCL ENTER
set name [sh_input string "Enter your name"]; ENTER
if {$name=="SH_CANCEL"} { return ""; } ENTER
return "Your Name: $name\nYour Name in UPPERCASE: [string toupper
$name]"; ENTER
}
```

An alternative is to use the @INPUT tag to place the user input into a Shorthand variable and embed the variable in the @INPUTTCL script. This example does the same thing as the previous example except that an @INPUT tag is used to get the name which is passed to a Tcl script through the “{name}” variable:

```
Your Name: {@INPUT(name) Enter your name:} ENTER
Your Name in UPPERCASE: {@INPUTTCL ENTER
string toupper "{name}" ENTER
}
```

Note that the double quotes (`""`) around `{name}` are necessary since `{name}` can contain spaces and the `string toupper` Tcl command expects only a single argument.

Use the @REM Tag to prevent the result of an @INPUT tag from being displayed.

If you enclose an @INPUT tag inside an @REM tag, the result of the @INPUT tag will not be inserted into your word processor. The following code is the same as the previous example except that only your name in uppercase is sent to your word processor:

```
{@REM ENTER
{@INPUT(name) Enter your name:} ENTER
}Your Name in UPPERCASE: {@INPUTTCL ENTER
string toupper "{name}" ENTER
}
```

Use 'run' instead of 'exec' to launch applications.

Using the standard Tcl `exec` command to launch an external program will lock up Shorthand since Shorthand cannot continue until the program that was launched is completed. The `exec` command also does not work with command-line calls (such as command.com and the `dir` command).

Instead of using **exec**, launch external programs with the `run` command. The `run` command is a Tcl extension command implemented by Shorthand and takes the same arguments as the **Run** command found by clicking on the Windows **START** button.

Here is an example that opens a file called MyFile.doc in Microsoft Word:

```
{@INPUTTCL ENTER
run "C:/Program Files/MSOffice/Word.exe" "MyFile.doc" ENTER
}
```

In Tcl scripts, you will need to convert all backslashes '\' to forward slashes '/' when specifying Windows path names.

The `run` command recognizes Windows registered file types. For example, the following code will open a web site in your default Internet browser

```
{@INPUTTCL ENTER
run "http://www.pcshorthand.com" ENTER
}
```

Delete unneeded variables.

Any Tcl variable declared in a Tcl script will stay in memory until you explicitly clear it. It is possible to run out of Windows memory and lock up your computer if the space occupied by variables grows too large. Use the Tcl `unset` command to delete a variable.

In the following example, the first @INPUTTCL tag reads a file of names and displays the names in sorted order. The second @INPUTTCL tag calls the Tcl `unset` command to destroy the variables created in the first @INPUTTCL tag:

```
{@INPUTTCL ENTER
  # open a file containing a list of names ENTER
  set f [open "C:/My Documents/namelist.txt"] ENTER
  # read the file's contents to a 'names' array ENTER
  # (assumes the file contains 1 name per line) ENTER
  while {[gets $f line] >=0 } { ENTER
    lappend nameList $line ENTER
  } ENTER
  # close the file ENTER
  close $f ENTER
  #  sort the list ENTER
  set nameList [lsort $nameList] ENTER
  # display the sorted list; one name per line ENTER
  return [join $nameList "\n"] ENTER
} {@INPUTTCL ENTER
  # clean up: destroy the variables used above ENTER
  unset nameList ENTER
  unset f ENTER
  unset line ENTER
}
```

Create global commands/variables in the SHORTHND.TCL file.

When Shorthand first starts up, Shorthand executes the contents in SHORTHND.TCL as a Tcl script. You can therefore introduce global commands or variables by placing them in the SHORTHND.TCL file.

For example, you can create a global variable for your name in SHORTHAND.TCL by adding the line:

```
set myName "Jane Smith"
```

And you can use the variable in your Shorthand text:

```
Sincerely, ENTER
{@INPUTTCL return $myName }
```

So if you give your Shorthand dictionary to someone else, that person can reuse your dictionaries by simply entering her name into the SHORTHND.TCL file instead of having to modify all the Shorthand entries that contains your name.

See also p. 101 for an example of how to define a new Tcl command that can be called from all your Tcl scripts.

Do not use Tcl commands that read/write to the console.

Shorthand does not create a Tcl console window so Tcl commands that read/write to the console (such as `puts "hello"`) will NOT work and will generate an error.

Use Tcl s backslash substitution to handle special characters in strings.

The characters **[ ]{ } "** *have special meaning in Tcl. If you need to include those characters as part of a string, you can do so by preceding the character with a backslash. For example, the Tcl command*

```
puts "\"hello\""
```

will print out hello (with double quotes) . Refer to a Tcl book for additional backslash substitution characters.

How to run Tcl scripts in Shorthand

Tcl scripts are embedded in your Shorthand code through the {@INPUTTCL} tag.

The syntax of the @INPUTTCL Tag is:

{@INPUTTCL *script*}

where *script* is a Tcl script and can contain any number of lines.

How are {@INPUTTCL} tags processed?

Shorthand evaluates the Tcl *script* through the Tcl interpreter and replaces the {@INPUTTCL} tag with the result of the script. The result of the script is either the argument of the Tcl `return` command or the result of the last line in the script. There are three special return values: "SH_BACK", "SH_CANCEL" and "SH_RESTART". If your Tcl script returns the string "SH_BACK", Shorthand will go back and display the previous @INPUT* tag. If the return value is "SH_CANCEL", Shorthand will abort the input session. A return value of "SH_RESTART" tells Shorthand to restart from the beginning.

When are {@INPUTTCL} tags processed?

Shorthand processes all {@INPUTTCL} tags before inserting any text into your word processor. This means that, once expansion has started, all Tcl scripts have already been evaluated.

Example problem

How do I write a Tcl script to type out today's date in the following format: "month/day/year"?

Solution

Create a dictionary entry with the following Shorthand code:

Text to Type:

```
{@INPUTTCL ENTER
   set t [clock seconds] ENTER
   clock format $t -format "%m/%d/%y" ENTER
}
```

How it works

Here's an explanation of the code in the Text to Type box:

`{@INPUTTCL`	Starts the INPUTTCL tag..
`set t [clock seconds]`	Gets the current time (in seconds) and stores it in the Tcl variable called "t".
`clock format $t -format "%m/%d/%y"`	Reads the time stored in variable **t**. and converts it to a string in the format "month/day/year". Since this is the last line of the script, the result of this line will be returned to Shorthand.
`}`	Closing brace for the @INPUTTCL tag. Be careful not to type an extra space after the '}".

How to run a Tcl script with a single keystroke

You can run a Tcl script with a single keystroke by creating a *File Shortcut*. Follow the procedure on p. 42 to create a file shortcut and specify the filename of the Tcl script in step 3. Note that the Tcl file must end with the extension ".tcl" so Shorthand knows the file contains a Tcl script.

A note on Tcl variables

Shorthand evaluates scripts in the same Tcl interpreter so any variables you define are global and can be used in other @INPUTTCL tags. In the following example, the second @INPUTTCL tag is able to read the current time from the `t` variable which was initialized in the first @INPUTTCL tag:

Text to Type:

```
Today's Date is {@INPUTTCL [ENTER]
set t [clock seconds] [ENTER]
clock format $t -format "%m/%d/%y" [ENTER]
}. [ENTER]
Today is {@INPUTTCL [ENTER]
clock format $t –format "%A" [ENTER]
}.
```

will result in something like:

Today s Date is 08/15/00.

Today is Tuesday.

How to launch applications with Tcl

The `run` Tcl command can be used to launch applications such as your word processor or web browser. The syntax of the `run` Tcl command is:

run ***filename***

Example problem

How do I use Shorthand to open my web browser to the http://www.pcshorthand.com web site?

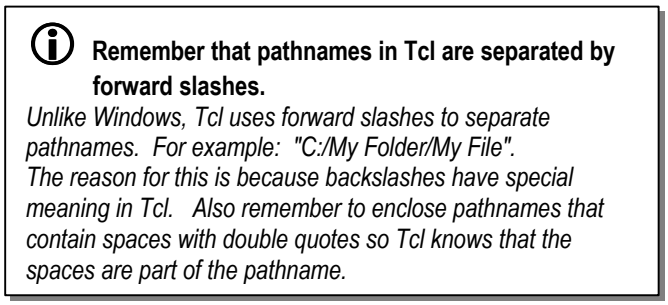
Solution

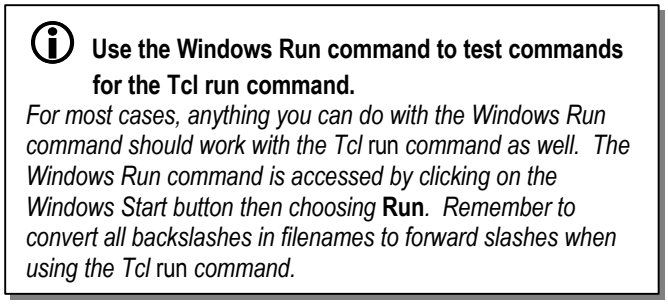
Create a new Shorthand entry with the following data:

Keyword: `web1`

Text to Type:

```
{@INPUTTCL ENTER
run "http:://www.pcshorthand.com" ENTER
}
```





How to simulate keystrokes with Tcl

You can use the `sh_screen key` command to simulate a keystroke from within a Tcl script. The syntax is:

sh_screen key *args*

where ***args*** is one or more keystroke codes (e.g. **Shift+F12** or **Ctrl+Alt+Space**) separated by spaces; the `sh_screen key` command is the Tcl equivalent of the {@KEY} tag (p. 68).

Example

See p. 108 for an example of how the use the `sh_screen key` command.

Notes

- Unlike the {@KEY} tag, sh_screen key is executed when Shorthand reads the Tcl script and not when the long form is expanded. You normally need a `sh_window pause` command (p. 108) before or after the command to make sure the key is interpreted correctly by your word processor.
- To simulate more than one keystroke, separate the keystroke codes with spaces. For example, the command to simulate the word `"Hello"` is:

```
sh_screen key "H" "e" "l" "l" "o"
```

Use the clipboard to "read" text from your word processor.

You can transfer text from your word processor to Shorthand through the clipboard. For example, the following code capitalizes the last word in your word processor:

```
{@INPUTTCL
# make sure Alt key is released
while {[sh_screen iskeydown alt} { };
sh_screen key ctrl+shift+left;  # select last word
sh_screen key ctrl+c;  # copy to clipboard
sh_window pause 100; # give time for clipboard copy
set s [sh_set clipboard]; # get text from clipboard
return [string totitle $s]; # capitalize and return
}
```

*Note: To make the above code work, create a dictionary entry (p. 24), copy the above code (exactly 9 lines) to the Text to Type box and assign a shortcut key (e.g. **Ctrl+F9**). To use the code, simply press the shortcut key (e.g. **Ctrl+F9**) after a word in your word processor.*

How to import text from another dictionary entry

The `sh_list` command can be used to find and read the text from another entry in the active dictionary.

Example problem

I already have two dictionary entries containing my first name and last name:

```
fname = John
lname = Smith
```

How do I create a new entry that will import `fname` and `lname` and form my full name?

Solution

Create a new Shorthand entry with the following data:

Keyword: `fullname`

Text to Type:

```
{@INPUTTCL
proc sh_expand {keyword} { ENTER
  set i [sh_list search $keyword]; ENTER
  if {$i >= 0} { ENTER
    return [sh_list text $i]; ENTER
  } else { ENTER
    return "";  # not found ENTER
  } ENTER
} ENTER
return "[sh_expand fname] [sh_expand lname]" ENTER
}
```

How it works

`proc sh_expand {keyword}`	Defines a new Tcl command called **sh_expand** which finds and return the text associated with $keyword.
`set i [sh_list search $keyword];`	Searches for $keyword in the active dictionary and sets i to the index number (an integer) of the entry.
`if {$i >= 0} {`	Tests if the $keyword was found: If the index number is greater than or equal to zero, the $keyword exists in the dictionary.
`return [sh_list text $i];`	Returns the text (long form) associated with the entry at index $i.

`} else {`
`  return "";`
`  }`
	If $i is negative, the $keyword was not found, so we just return a blank string.

`return "[sh_expand fname] [sh_expand lname]"`	Invokes the sh_expand command to fetch the text of **fname** and **lname** and returns the combined string to be typed out by Shorthand.

How to make sh_expand available to all Tcl scripts

If you define the `sh_expand` proc in the SHORTHND.TCL file (see p. 95), the `sh_expand` command will be available to all your Tcl scripts. That is, if your SHORTHND.TCL file contains:

```
proc sh_expand {keyword} { ENTER
  set i [sh_list search $keyword]; ENTER
  if {$i >= 0} { ENTER
    return [sh_list text $i]; ENTER
  } else { ENTER
    return "";  # not found ENTER
  } ENTER
} ENTER
```

The Text to Type of the above example can be simply:

```
{@INPUTTCL ENTER
return "[sh_expand fname] [sh_expand lname]" ENTER
}
```

How to perform arithmetic computations

The `expr` Tcl command can be used to evaluate math expressions. The syntax is:

expr ***expression***

where *expression* is a mathematical expression such as `(1 + 6)*1.08;` refer to a Tcl book or the Tcl web site (p. 91) for detailed documentation on the `expr` command.

Example problem

How do I display a dialog box asking for the user's birth date then compute the user's age from the user's input?

Solution

Create a Shorthand entry with the following data:

Keyword: `age1`

Text to Type:

```
{@INPUTTCL
# Get birthdate [ENTER]
set s [sh_input date "Enter your birth date:" ""]; [ENTER]
if {$s == "SH_CANCEL"} {return ""} [ENTER]
# compute age [ENTER]
set date_seconds [clock scan $s]; [ENTER]
set now_seconds [clock seconds]; [ENTER]
set age_seconds [expr $now_seconds - $date_seconds]; [ENTER]
set seconds_per_year [expr 365 * 24 * 60 * 60]; [ENTER]
set age_years [expr $age_seconds/$seconds_per_year]; [ENTER]
# return age [ENTER]
return "Your age is: $age_years"; [ENTER]
}
```

Test the entry

1) Make sure Shorthand is running and minimized (click Shorthand's Hide button).

2) Bring up your word processor and create or open a document.

3) In your word processor, type: `age1` SPACE

4) Shorthand displays a date dialog box asking for a birth date. Enter your date of birth and click OK.

5) Shorthand inserts your age in your word processor.

How it works

`set s [sh_input date "Enter your birth date:" ""];`	Displays date dialog box and returns the result in variable **s.**
`if {$s == "SH_CANCEL"} {return ""}`	Checks if the user had clicked the CANCEL button.
`set date_seconds [clock scan $s];`	Converts the date string in s to time in seconds.
`set now_seconds [clock seconds];`	Gets the current time in seconds.
`set age_seconds [expr $now_seconds - $date_seconds];`	Computes age (in seconds) by subtracting user's birth date from today's date.
`set seconds_per_year [expr 365 * 24 * 60 * 60];`	Converts age in seconds to age in years.
`return "Your age is: $age_years";`	Returns the result.

Tcl supports advanced mathematical expressions.
The Tcl expr *command supports advanced mathematical functions such as sines and cosines, logarithms, square roots and random numbers. Consult a Tcl book for more details on the* expr *command.*

How to enable/disable AutoReplace with a single keystroke

You can turn on/off AutoReplace with the `sh_set autoreplace` Tcl command.

The syntax is:

sh_set autoreplace *value*

where ***value*** is either **1** to enable AutoReplace or **0** to turn off AutoReplace.

Example problem

How do I turn on/off AutoReplace with single keystroke?

Solution

We will use the **toggle_autoreplace.tcl** script that comes with Shorthand; you should find this file in your Shorthand folder. If you can't locate it, you can use NOTEPAD to create a file called toggle_autoreplace.tcl that contains the following 5 lines:

```
if {[sh_set autoreplace ]} { ENTER
  sh_set autoreplace 0; ENTER
} else { ENTER
  sh_set autoreplace 1; ENTER
}
```

Follow these steps to create a shortcut key to launch the toggle_autoreplace.tcl script:

1) Choose **Shortcuts⇨Edit** . The **3. File Shortcuts** tab in Shorthand Preferences window appears.

2) Click the **Add** button. The Specify a File Shortcut window appears.

3) Click the Browse button and open the **toggle_autoreplace.tcl** file.

4) In the Specify a File Shortcut window, click the Shortcut button and press **Shift+Ctrl+A** (or you can specify another keyboard shortcut if you want).

5) In the Specify a File Shortcut window, enter "Toggle autoreplace" in the **Description** box.

6) In the Specify a File Shortcut window, click OK to save your entry. The Preferences window reappears, click OK again to close the window and return to Shorthand's Main window.

Test the entry

1) Make sure Shorthand is running and minimized (click Shorthand's Hide button).

2) Start your word processor.

3) In your word processor, press **Shift+Ctrl+A** to switch Shorthand's AutoReplace on and off. When AutoReplace is off, Shorthand will not expand abbreviations as you type. If you enabled Shorthand's Suggestion Window, you should notice the Suggestion Window appear when AutoReplace is on and disappear when AutoReplace is turned off.

How it works

The **Shortcuts⇨Edit** command allows you to assign a keyboard shortcut to run a Tcl script stored in a file. Here's an explanation of the contents of the toggle_autoreplace.tcl file:

`if {[sh_set autoreplace ]} {`	Tests if AutoReplace is currently on.
`sh_set autoreplace 0;`	If AutoReplace is currently on, turn it off

`} else {`
`  sh_set autoreplace 1;`
`}`
	else turn it on.

How does Shorthand know if a file contains a Tcl script?

Shorthand looks at the filename's extension to determine the contents of a file. Files that contain Tcl scripts end with the extension ".tcl". Files that contain Shorthand dictionaries have an extension of ".spf".

How to show/hide the Suggestion Window with a single keystroke

You can show or hide the Suggestion Window with the `sh_set suggestion_window` command.

The syntax is:

sh_set suggestion_window *value*

where ***value*** is either **1** to enable AutoReplace or **0** to turn off AutoReplace.

Example problem

How do I show or hide the Suggestion Window with a single keystroke?

Solution

We will use the **toggle_suggestion_window.tcl** script that comes with Shorthand; you should find this file in your Shorthand folder. If you can't locate it, you can use NOTEPAD to create a file called **toggle_suggestion_window.tcl** that contains the following 5 lines:

```
if {[sh_set suggestion_window ]} { ENTER
   sh_set suggestion_window 0; ENTER
} else { ENTER
   sh_set suggestion_window 1; ENTER
}
```

Follow these steps to create a shortcut key to launch the toggle_suggestion_window.tcl script:

1) Choose **Shortcuts⇨Edit**. The **3. File Shortcuts** tab in Shorthand Preferences window appears.

2) Click the **Add** button. The Specify a File Shortcut window appears.

3) Click the Browse button and open the **toggle_suggestion_window.tcl** file.

4) In the Specify a File Shortcut window, click the Shortcut button and press **Ctrl+Shift+S** (or you can specify another keyboard shortcut if you want).

5) In the Specify a File Shortcut window, enter "Toggle suggestion window" in the **Description** box.

6) In the Specify a File Shortcut window, click OK to save your entry. The Preferences window reappears, click OK again to close the window and return to Shorthand's Main window.

Test the entry

1) Make sure Shorthand is running and minimized (click Shorthand's Hide button).

2) Start your word processor.

3) In your word processor, press **Ctrl+Shift+S** to switch Shorthand's Suggestion Window on and off. Note that AutoReplace must be enabled for the Suggestion Window to work.

How it works

The **Shortcuts⇨Edit** command allows you to assign a keyboard shortcut to run a Tcl script stored in a file. Here's an explanation of the contents of the toggle_suggestion_window.tcl file:

`if {[sh_set suggestion_window ]} {`	Tests if AutoReplace is currently on.
`  sh_set suggestion_window 0;`	If AutoReplace is currently on, turn it off

`} else {`
`  sh_set suggestion_window 1;`
`}`
	else turn it on.

Use the Suggestion Window to debug Shorthand.
The first line of the Suggestion Window always displays the contents of Shorthand's internal keyboard buffer. If you experience problems with Shorthand, you can examine what Shorthand thinks you are typing and compare it with what you are actually typing. This helps you determine if Shorthand is working properly.

How to add delays within a Tcl script

You can use the `sh_window pause` command to temporarily pause a Tcl script. The syntax is:

sh_window pause *value*

where ***value*** is the minimum delay time in milliseconds.

Unlike the {@PAUSE} tag (p. 72), the `sh_window pause` command is evaluated immediately when Shorthand reads the Tcl script and not when Shorthand writes the Text to Type to your word processor. So placing a `sh_window pause` command between {@KEY} tags will have no effect; the `sh_window pause` command should be used only to insert delays inside a Tcl script.

You normally need to use the `sh_window pause` command after a `sh_screen key` (p. 99) or `run` (p. 98) command to allow the computer to process the command.

Example problem

What is the Tcl equivalent of the example on p. 72 to create a keyboard shortcut to move the cursor to the next occurrence of the word "INPUT:" in a Microsoft Word document?

Solution

Create a new Shorthand entry (p. 24) with the following data:

Keyword: `find input1` Shortcut: `Ctrl+Alt+I`

Text to Type:

```
{@INPUTTCL
while {[sh_screen iskeydown alt]} { } ENTER
while {[sh_screen iskeydown ctrl]} { } ENTER
sh_screen key "alt+e";  sh_window pause 100  ENTER
sh_screen key "f";  sh_window pause 500  ENTER
sh_screen key "I";  sh_window pause 100  ENTER
sh_screen key "N";  sh_window pause 100  ENTER
sh_screen key "P";  sh_window pause 100  ENTER
sh_screen key "U";  sh_window pause 100  ENTER
sh_screen key "T";  sh_window pause 100  ENTER
sh_screen key ":";  sh_window pause 100  ENTER
sh_screen key "alt+f";  sh_window pause 100  ENTER
return ""  ENTER
}
```

Test the entry

1) Make sure Shorthand is running and minimized (click Shorthand's Hide button).

2) Bring up Microsoft Word and create or open a document that contains several instances of the word "INPUT:".

3) In Microsoft Word, press **Ctrl+Alt+I**.

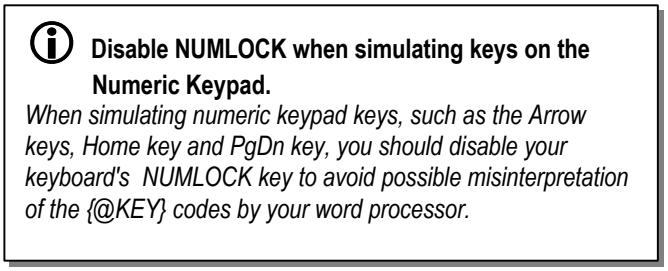
You should see Word bring up the Find dialog box, search for the next instance of "INPUT:" then close the Find dialog box.

How it works

Unlike the {@KEY} tag, which is executed after Shorthand has returned control to the word processor, the `sh_screen key` command is executed immediately. The first two lines use the `sh_screen iskeydown` command to wait until the user releases the **Ctrl** and **Alt** keys; without this pause, the Ctrl and Alt keys may still down when `sh_screen key "alt+e"` is executed which will be interpreted by your word processor as **Ctrl+Alt+E** instead of **Alt+E**.

The `sh_window pause 100` commands are used to insert short 100 millisecond pauses after every keystroke to give Microsoft Word time to process each keystroke.

The script ends with a `return ""` which returns a blank string to your word processor.



How to convert text to Title Case

You can use Tcl's `string` command to process text strings. The syntax of the `string` command is:

string *option args*

where ***option*** is one of many operations supported by Tcl; example options are:

- **tolower**, **totitle** and **toupper** to change the case of ***args***.
- **trim**, **trimright** and **trimleft** to remove spaces from ***args***.
- **length** to determine the number of characters in ***args*.**
- **index** to return the character at a specified position.

Refer to a Tcl book or the Tcl web site (p. 91) for detailed documentation on the **string** command.

Example problem

How do I convert a sentence to title case (i.e. capitalize the first character of each word)?

Solution

Create a new Shorthand entry (p. 24) with the following data:

Keyword: `totitle`

Text to Type:

```
{@INPUTTCL ENTER
set s [sh_input string "Enter sentence"] ENTER
if {$s == "SH_CANCEL"} { return "";} ENTER
set s [string totitle $s]; ENTER
regsub -all {\s\w} $s {[string toupper {\0}]} s2 ENTER
return [subst $s2]; ENTER
}{@NOSPACE}
```

Test the entry

1) Make sure Shorthand is running and minimized (click Shorthand's Hide button).
2) Bring up your word processor and type: `totitle` SPACE

Shorthand displays a dialog box asking you to enter a sentence.

3) In the dialog box, type: `medical report` ENTER

Shorthand inserts Medical Report into your word processor.

How it works

`set s [sh_input string "Enter sentence"]`	Displays an input dialog box for the user to enter a sentence.
`if {$s == "SH_CANCEL"} { return "";}`	If the user clicked the Cancel button, a blank string is returned to your word processor.
`set s [string totitle $s];`	Use Tcl's **string totitle** command to capitalize the first letter in the sentence; the remaining characters will be in lower case.
`regsub -all {\s\w} $s {[string toupper {\0}]} s2`	Use Tcl's **regsub** command to capitalize each character that comes after a space.
`return [subst $s2];`	Returns the processed string to your word processor. The subst command is necessary to evaluate the **string toupper** command in the previous line.
`}{@NOSPACE}`	The {@NOSPACE} tag (p. 85) prevents extra spaces from appearing in the final output.

How to get user input with Tcl

You can use the `sh_input` command to ask for user input; the syntax is:

sh_input ***option prompt arg ?arg ...?***

where ***option*** specifies the type of dialog box and ***prompt*** is a string you wish to be displayed at the top of the dialog box. The return value is the user input if the user clicks the OK button. The return value is the string "SH_CANCEL" if the user clicks on the CANCEL button.

The legal options are:

`sh_input string` *prompt ?defaultValue*?	Displays an input dialog window with a single line text box. The text box is initialized with *defaultValue* and the user may type anything in the text box.
`sh_input memo` *prompt ?defaultValue?*	Same as `sh_input string` except that a multi line text box is shown in the dialog window.
`sh_input date` *prompt ?defaultValue?*	Same as `sh_input string` except that a date input box is shown in the dialog window.
`sh_input open` *prompt ?defaultValue?*	Same as `sh_input string` except that an Open File dialog window is shown.
`sh_input save` *prompt ?defaultValue?*	Same as `sh_input string` except that a Save As dialog window is shown.
`sh_input list` *prompt {listItems} ?{defaultSelections}?*	Displays a dialog window with a multi-selection pick list. *listItems* is a list of strings to display in the pick list. *defaultSelections* are the index numbers of items in *listItems* that you want selected when the dialog window is first displayed. The return value is a list of index numbers of selected items if the user clicked the OK button. The return value is SH_CANCEL if the cancel button was clicked.
`sh_input list1` *prompt {listItems} ?{defaultSelections}?*	Same as `sh_input list` except that only item may be selected.
`sh_input msg` *prompt message*	Displays a message to the user.
`sh_input spellcheck` *prompt text*	Displays a multi-line dialog window and then spell checks text. Same as executing `sh_input memo` and clicking on the Spell Check button.

Example problem

How do I use Shorthand to type out the following sentence?

Upon entry, the patient was suffering from: (list of symptoms).

where the list of symptoms can be one or more of:

Dizziness, Dry Mouth, Headache, Abdominal Pain, Fever

If the patient has more than one symptom, the symptoms are to be separated by slashes "/".

Solution

Create a Shorthand entry with the following data:

Keyword: `symps1`

Text to Type:

```
{@INPUTTCL
set items {"Dizziness" "Dry Mouth" "Headache"  "Abdominal Pain"
"Fever"}; ENTER
set s [sh_input list "Choose symptoms that apply" $items]; ENTER
if {$s == "SH_CANCEL"} {return ""} ENTER
set symptoms ""; ENTER
foreach i $s { ENTER
   lappend symptoms [lindex $items $i]; ENTER
} ENTER
if {[llength $symptoms] > 1} { ENTER
   set symptoms [join $symptoms "/"]; ENTER
} else { ENTER
  set symptoms [join $symptoms]; ENTER
} ENTER
return "Upon entry, the patient was suffering from:  $symptoms";
ENTER
}
```

Test the entry

1) Make sure Shorthand is running and minimized (click Shorthand's Hide button).

2) Bring up your word processor and create or open a document.

3) In your word processor, type: `symps1` SPACE

4) Shorthand displays a dialog box asking for the symptoms with a list of choices. Select one or more items and click OK.

5) Shorthand inserts the selected items in your word processor.

How it works

`set items {"Dizziness" "Dry Mouth" "Headache" "Abdominal Pain" "Fever"}`	Creates a Tcl list variable initialized with the 5 symptoms.
`set s [sh_input list "Choose symptoms that apply" $items];`	Calls sh_input to display a pick list dialog box and returns the result to variable **s**. The result is a list of indices into **items**.
`if {$s == "SH_CANCEL"} {return ""}`	If the user click CANCEL, return an empty string (i.e. Shorthand inserts nothing in your word processor)..
`set symptoms "";`	Creates a Tcl variable called **symptoms**. We will use this variable to hold the symptoms selected by the user.

`foreach i $s {`
`lappend symptoms [lindex $items $i]`
`}`
	This loops through each element in the **s** list variable and converts each numerical index in **s** to a list of symptom strings.

`if {[llength $symptoms] > 1} {`
`set symptoms [join $symptoms "/"];`
`}`
	If **symptoms** has more than one element, join the list of symptoms into a single string with each element separated by slashes "/."

`else {`
`set symptoms [join $symptoms];`
`}`
	If symptoms has only one element then just convert the list of symptoms into a string.

`return "Upon entry, the patient was suffering from:  $symptoms";`	Returns the result.

How to export a dictionary to a comma-delimited text file

If Shorthand's built-in Print to File command (p. 21) does not export your word list in the right format you can write own Tcl script to dump the contents of the dictionary in whatever format you need.

Example problem

How do I export my word list to a comma-delimited text file with the strings enclosed in double quotes? If the string contains a double quote, the double quote should be preceded by a backslash (\). The output should look something like:

"keyword", "this is the long form" ENTER

"t1", "\"this long form is enclosed in double quotes\"" ENTER

"t2", "this long form has ENTER

two lines"

Solution

Create a file called **sh_print_dictionary.tcl** with the following Tcl code (the file can also be downloaded from http://www.pcshorthand.com/book):

```
proc _print_dictionary { } {
  # get complete pathname of current dictionary.
  set dictionaryPathName [sh_file];

  # extract rootname of dictionary
  set dictionaryName [file rootname $dictionaryPathName];

  # ask for a filename with the default name being "dictionaryName.TXT"
  set filename [sh_input save "Enter filename to save the word list"
"${dictionaryName}.txt"];
  if {$filename == "SH_CANCEL"} {return;}

  # open file for writing
  set f [open $filename w];
  set n [sh_list length];  # get number of entries in active dictionary
  set count 0;

  for {set i 0} {$i < $n} {incr i} {
    if {[sh_list readonly $i]} {
      continue; # Skip READONLY entries which are NOT part of the main file
    }
    set keyword [sh_list keyword $i];  # get short form
    set longform [sh_list text $i];  # get long form
```

```
    # insert a backslash in front of all double quotes
    regsub -all {"} $keyword {\\"} keyword;
    regsub -all {"} $longform {\\"} longform;

    # remove carriage return (ASCII code 13) characters from long form since
    # the puts command will automatically insert a carriage return
    # before every linefeed (ASCII code 10) character.
    regsub -all {\r} $longform "" longform;

    # write to file as comma-delimited, double-quoted strings
    puts $f "\"${keyword}\", \"$longform\"";
    incr count;
  }
  # close the file
  close $f;

  # display user message
  sh_input msg ""  "$count entries were written to: $filename";

  # The following lines work around this bug in Sh851 and earlier by
  # redisplaying the main window if it is visible
  if {[sh_window show] == 1} {
    # we use "catch" to suppress focus error messages
    # which could occur if you use Win95.
    catch {sh_window show 1;}
  }
  return;
}

# run the proc
_print_dictionary;
rename _print_dictionary ""; # remove proc from memory.
return; # all done
```

How to run the script

There are two ways to run the script:

METHOD #1: From Shorthand's Shortcut Menu

1. Place the sh_print_dictionary.tcl file in your Shorthand folder.
2. From Shorthand's **Shortcut** menu, choose **Edit**.
3. Click the **Add** button.
4. Click the **Browse** button and select the sh_print_dictionary.tcl file.
5. In the Specify a File Shortcut box, specify a keyboard shortcut (e.g. **Shift+Ctrl+P**)
6. In the Description box, type in a short description (e.g. "Print").
7. Click OK as necessary to return back to Shorthand main window.
8. You can now run the script by invoking the keyboard shortcut or by choosing "Print" from Shorthand's Shortcut menu.

METHOD #2: From Shorthand's File Menu

1. From Shorthand's **File** menu, choose **Tcl Script**.
2. Click the **Open** button and load in the sh_print_dictionary.tcl file.
3. Click the **Eval Script** button.

How it works

The script starts by calling the `sh_input save` command to ask for the destination filename from the user. The script then loops through every entry in the active dictionary and calls the `sh_list keyword` and `sh_list text` commands to read the short and long forms of each dictionary entry. The `regsub` command is used to insert a backslash in front of every double quote. Before writing the long form to the text file, the `regsub` command is called again to strip out carriage returns from the long form; this is necessary because, on the Windows platform, Tcl's `puts` command automatically inserts a carriage return (ASCII code 13) character in front of every linefeed (ASCII code 10) character.

How to import text from a comma-delimited file

This is the opposite problem of the script in the previous section (p. 115). The example script below shows how you can use Tcl to read a comma-delimited word list into a Shorthand dictionary. The contents of comma-delimited file should look like:

"keyword", "this is the long form" ENTER

"t1", "\"this long form is enclosed in double quotes\"" ENTER

"t2", "this long form has ENTER

two lines"

☞ ***WARNING***: This script is intended for illustration purposes only. The script modifies your dictionaries and can corrupt data if not used correctly. Back up your Shorthand dictionaries before performing executing the script as you can easily mess up your word lists if you make a mistake.

Create the Tcl script file

Create a file called **sh_import_text.tcl** with the following Tcl code (the file can also be downloaded from http://www.pcshorthand.com/book):

```
# Proc to search and replace {\"} with "{@REM DQ}"
proc _removeDQ {s} {
  set searchFor {\\"};
  set replaceWith "{@REM DQ}";
  regsub -all $searchFor $s $replaceWith result;
  return $result;
}

# Proc to search and replace "{@REM DQ} with "
proc _restoreDQ {s} {
  set searchFor "{@REM DQ}";
  set replaceWith {"};
  regsub -all $searchFor $s $replaceWith result;
  return $result;
}

# Proc to read an entry. Returns 1 if successful. Returns 0 at end-of-file.
proc _get_entry { file shortformVar longformVar } {
  upvar $shortformVar shortform;
  upvar $longformVar longform;
  set $shortform "";
  set $longform "";
  while {1} {
    if {[gets $file line] < 0} {
      return 0; # end-of-file (no more entries)
    }
```

```
    # temporarily hide the \" codes in $line
    set line [_removeDQ $line];

    # convert $line to a list with " as the separator
    set s [split $line {"}];

    # a valid line should have at least 3 double-quotes (or split
    # into at least 4 entries); we skip lines that are invalid
    set n [llength $s];
    if {$n >= 4} {
      break;
    }
  }

  # Short form is the second list entry
  set shortform [lindex $s 1];
  set shortform [_restoreDQ $shortform]; # restore double quotes

  # Long form is the 4th list entry
  set longform [lindex $s 3];

  # if $line contains 4 or more double-quotes, we have
  # the complete long form (long form is a single line)
  if {$n > 4} {
    # restore double quotes and exit
    set longform [_restoreDQ $longform];
    return 1;
  }

  # Get the  multi-line long form by reading lines until
  # we encounter the terminating double-quote.
  while {1} {
    if {[gets $file line] < 0} {
      break;
    }
    set line [_removeDQ $line];
    set i [string first {"} $line];
    if {$i >= 0} {
      set s [string range $line 0 [expr $i - 1]];
      append longform "\r\n" $s;
      break;
    } else {
      append longform "\r\n" $line;
    }
  }
  # restore double quotes and exit
  set longform [ _restoreDQ $longform];
  return 1;
}
```

```
# This is the main proc
proc _import_text {} {
  # Get input filename
  set infile [sh_input open "Select comma-delimited text file to import"];

  if {[string equal $infile "SH_CANCEL"] == 1} {return 0};

  # Get output .spf filename
  set spffile "[file rootname $infile].spf";
  set spffile [sh_input save "Enter destination Shorthand dictionary (you can
specify a new file)" $spffile];
  if { [string equal $spffile "SH_CANCEL"] == 1} {return 0};

  # Make sure links are disabled (a bug in SH851 will display the
  # Preferences dialog box if the links list is empty; if this happens,
  # just close the Preferences box by clicking OK.
  sh_set links 0;

  # open the input file
  set f [open $infile r];

  # Open the target .spf file
  sh_file open $spffile;

  # start the conversion
  set i 0;
  while {1} {
    set shortform "";
    set longform "";
    set result [ _get_entry $f shortform longform ];
    if {$result == 0} {
      break;
    }
    # write entry to dictionary
    sh_list update $shortform $longform
    incr i;
  }

  # display result
  sh_input msg "Conversion Completed!" "$i entries have been imported";
  return;
}

# run our script
_import_text;

# remove procs from memory
rename _import_text {}
rename _get_entry {}
rename _removeDQ {}
rename _restoreDQ {}
return "";
```

How to run the script

There are two ways to run the script:

METHOD #1: From Shorthand's Shortcut Menu

1. Place the sh_import_text.tcl file in your Shorthand folder.
2. From Shorthand's **Shortcut** menu, choose **Edit**.
3. Click the **Add** button.
4. Click the **Browse** button and select the sh_import_text.tcl file.
5. In the Specify a File Shortcut box, specify a keyboard shortcut (e.g. **Shift+Ctrl+I**)
6. In the Description box, type in a short description (e.g. "Import text").
7. Click OK as necessary to return back to Shorthand main window.
8. You can now run the script by invoking the keyboard shortcut or by choosing "Import text" from Shorthand's **Shortcut** menu.

There's a bug in Shorthand 8.51 and earlier in that the Preferences window may appear when you call the `sh_set links` command and your Dictionary links list is empty; if the Preferences window appears, just click on CANCEL to close it and the script will continue.

METHOD #2: From Shorthand's File Menu

1. From Shorthand's **File** menu, choose **Tcl Script**.
2. Click the **Open** button and load in the sh_export_text.tcl file.
3. Click the **Eval Script** button.

How it works

After calling the `sh_input open` and `sh_input save` commands to get the input file and destination Shorthand dictionary, the script calls the `_get _entry` proc to read the short and long forms from the input file. The `_get_entry` proc parses the input strings by calling Tcl's `split` command (with the double quote character as the separator) to split the input string into a list; the short form will be the second list element and the long form will be the fourth list element. To prevent the `split` command from processing double quotes that are really part of the short and long forms, the `_removeDQ` proc is used to temporarily replace every occurrence of the string `\"` to the string "{@REM DQ}". The `_restoreDQ` proc is used to restore the double quotes just before exiting the `_get_entry` proc. The `sh_list update` command is finally called to add the short and long forms to the active dictionary.

How to search with pattern matching

The following script is an example of a Tcl script that performs a search with pattern matching of the Text to Type field of each dictionary entry. When the search is completed, Shorthand brings up the Windows WORDPAD application with a list of the Keywords that contain the specified pattern. This script demonstrates how to use Tcl to:

- Display a dialog box to fetch an input string from the user.
- Loop through each dictionary entry to search for a particular string pattern using the `string match` command.
- Write to a text file.
- Launch an external application (WORDPAD) to display the text file.

Create the Tcl script file

Create a file called **sh_pattern_match.tcl** with the following Tcl code (the file can also be downloaded from http://www.pcshorthand.com/book):

```
proc _run_pattern_match {} {
  # display dialog boxes to get user input
  while {1} {
    set pattern [sh_input string "Enter pattern to search for" ""]
    if {$pattern==""} {continue;}
    if {$pattern=="SH_CANCEL"} {return}
    break
  }
  set outfile [sh_input save "Save results to" "search results.txt"]
  if {$outfile=="SH_CANCEL"} {return}
  set f [open $outfile w];
  puts $f "*** Searching for: \"$pattern\"";
  puts $f "";
  puts $f "Keyword";
  puts $f "-------";

  #--- Loop through the dictionary
  set count 0;
  set n [sh_list length];
  for {set i 0} {$i < $n} {incr i} {
    set s [sh_list text $i];  # get TEXT TO TYPE
    if {[string match -nocase $pattern $s]} {
      # write keyword name to file
      puts $f "[sh_list keyword $i]";
      incr count;
    }
  }

  puts $f "";
```

```
  puts $f "*** Found: $count entries";
  close $f; #close the file
  run "WordPad" "\"$outfile\"";  # bring up the file in WordPad
}

_run_pattern_match;

# The following lines work around a display bug in Sh8.51 and earlier
if {[sh_window show] == 1} {
  catch {sh_window show 1};
}
return "";
```

How to run the script

There are two ways to run the script:

METHOD #1: From Shorthand's Shortcut Menu

1. Place the sh_pattern_match.tcl file in your Shorthand folder.
2. From Shorthand's **Shortcut** menu, choose **Edit**.
3. Click the **Add** button.
4. Click the **Browse** button and select sh_pattern_match.tcl.
5. In the Specify a File Shortcut box, specify a keyboard shortcut (e.g**. Shift+Ctrl+S**)
6. In the Description box, type in a short description (e.g. "Search").
7. Click OK as necessary to return back to Shorthand main window.
7. You can now run the script by invoking the keyboard shortcut or by choosing **Search** from Shorthand's **Shortcut** menu.

METHOD #2: From Shorthand's File Menu

1. From Shorthand's **File** menu, choose **Tcl Script**.
2. Click the **Open** button and load in sh_pattern_match.tcl.
3. Click the **Eval Script** button.

How it works

The script loops through every entry in the active dictionary and uses the `sh_list text` command to get the long form. The long form is searched using Tcl's `string match` command which performs a pattern match of the entire long form. For example:

- The search pattern "min*" will match entries whose text BEGINS WITH the string "min" (e.g. "minimum", "Minnesota").
- The search pattern "*min*" will match entries whose text CONTAINS the string "min" (e.g. "unminted", "minimum", "vermin").
- The search pattern "*min" will match entries whose text ENDS WITH the string "min" (e.g. "vermin").

You can get more information on the `string match` command from any Tcl book or from the Tcl documentation at http://www.scriptics.com.

How to perform a global search and replace

The following script is an example of a complex Tcl script to perform a search and replace of words in your dictionaries. The script displays a pick list of the dictionaries that appear in your Dictionary tabs; you can select which file(s) to perform the search and replace.

This script demonstrates how to use Tcl to:

- Read the names of the dictionary tabs from the PAGES.CFG file in the Shorthand program folder and display the files in a pick list.
- Display a pick list of search options to the user.
- Loop through each dictionary entry and use Tcl's `regsub` command to do a search and replace.

WARNING: This script is intended for illustration purposes only. The script modifies your dictionaries and can corrupt data if not used correctly. Back up your Shorthand dictionaries before performing executing the script as you can easily mess up your word lists if you make a mistake.

Create the Tcl script file

Create a file called **sh_global_search.tcl** with the following Tcl code (the file can also be downloaded from http://www.pcshorthand.com/book):

```
# Proc that returns a list of filenames shown in the Dictionary tabs
# on Shorthand's Main Window.
# The names of the tabs are kept in the PAGES.CFG file
# in the Shorthand program folder.
proc sh_readPagesCfg {} {
  # folder where PAGES.CFG is stored
  set shCfgPath [sh_window cfgdir];
  # open PAGES.CFG
  set f [open "${shCfgPath}PAGES.CFG"];
  set result "";
  # Read contents of PAGES.CFG line by line
  while {[gets $f line] >= 0} {
    # convert "\" to "/"
    regsub -all {\\} $line "/" line;
    # add pathname if necessary
    if {[file dirname $line] == "."} {
      set line $shCfgPath$line;
    }
    lappend result $line;
  }
  return $result;
}
```

```
# This proc does the search and replace
proc _search_replace {old_word new_word text_var bWholeWordsOnly bCaseSensitive}
{
  # use Tcl's regsub command to search and replace $word in $text_var
  upvar $text_var text;
  set bMatch 0;
  if {$bCaseSensitive!=1} {
    set noCase "-nocase";
  } else {
    set noCase "-all";
  }
  if {$bWholeWordsOnly==1} {
    # replace $old_word if it occurs at the start of $text
    if {[regsub $noCase -all "${old_word}(?=\\W)" $text  $new_word text]} {
      set bMatch 1;
    }
    # replace $old_word if it occurs at the end of $text
    if {[regsub $noCase -all "(?=\\W)${old_word}" $text  $new_word text]} {
      set bMatch 1;
    }
    # replace $old_word if it occurs inside $text
    if {[regsub $noCase -all "(?=\\W)${old_word}(?=\\W)" $text $new_word text]}{
      set bMatch 1;
    }
  } else {
    set bMatch [regsub $noCase -all $old_word $text $new_word text];
  }
   return $bMatch;
}

proc _run_search {} {
  #--- Get the names of dictionaries in PAGES.CFG
  set files [sh_readPagesCfg];

  #--- Make sure active dictionary is in the list
  set currentFileIndex [lsearch $files [sh_file]];
  if {$currentFileIndex >= 0} {
    set defaultSelections $currentFileIndex;
  } else {
    set files [linsert $files 0 [sh_file]];
    set defaultSelections {0};
  }

  # Allow user to select which files to search
  set selections [sh_input list "Select input files" $files $defaultSelections];
  if {$selections == "SH_CANCEL"} return;

  # remove leading braces from $selections (needed only for SH8.51 or earlier).
  regsub "^{" $selections "" selections;
  regsub "}$" $selections "" selections;

  # create hex map
  for {set i 0} {$i < 256} {incr i} {
    set c [format %c $i];
```

```
  set hexMap($c) "[format \\x%02X $i]";
}
# display dialog boxes to get user input
while {1} {
  set searchFor [sh_input string "Enter word to search for" ""]
  if {$searchFor==""} {continue;}
  if {$searchFor=="SH_CANCEL"} {return}
  break;
}
# convert $searchFor to hex string so regsub will search
# for the literal characters
set searchForHex "";
set n [string length $searchFor];
for {set i 0} {$i < $n} {incr i} {
  set searchForHex $searchForHex$hexMap([string range $searchFor $i $i]);
}

set replaceWith [sh_input string "Enter replacement string" ""];
if {$replaceWith=="SH_CANCEL"} {return}
set sOptions [sh_input list "Select search options" {
   "Whole words only" "Case sensitive" "(leave this box checked)"} {0 1 2}];
if {$sOptions=="SH_CANCEL"} {return};

#--- Set up the search options
# First remove leading braces (needed only for SH8.51 or earlier).
regsub "^{" $sOptions "" sOptions;
regsub "}$" $sOptions "" sOptions;
set bCaseSensitive 0;
set bWholeWordsOnly 0;
if {[lsearch $sOptions "0"] >= 0} { set bWholeWordsOnly 1}
if {[lsearch $sOptions "1"] >= 0} { set bCaseSensitive 1}

#--- remember current file
set currentFile [sh_file];
set result "";

#--- Loop through each dictionary
foreach i $selections {
  set file [lindex $files $i];
  # skip non-existent files
  if { [file exists $file] == 0} {
    lappend result [list $file "(Error: cannot open file)"];
    continue;
  }

  # open file
  sh_file open $file;

  # perform search and replace
  set n [sh_list length];
  set count 0;
  for {set i 0} {$i < $n} {incr i} {
    if {[sh_list readonly $i]} {
      continue; # Skip READONLY entries which are NOT part of the main file
    }
```

```
      set s [sh_list text $i];  # get TEXT TO TYPE
      if {[_search_replace $searchForHex $replaceWith s bWholeWordsOnly
bCaseSensitive]} {
        sh_list text $i "$s";  # modify TEXT TO TYPE
        incr count;
      }
    }
    lappend result [list $file "Modified $count entries"];
  }

  # reopen current file
  sh_file open $currentFile;
  return $result;
}

set result [_run_search];
if {[llength $result] > 0} {
  sh_input msg "Search and Replace Results" [join $result "\n"];
}

# There's a bug in Sh8.51 and earlier in that the Shorthand main window
# remains minimized if you use the sh_input command through
# the File | Tcl Scripts menu or Shortcut menu.  The following lines work
# around this bug by redisplaying the main window if it is visible
if {[sh_window show] == 1} {
  catch {sh_window show 1};
}
return "";
```

How to run the script

There are two ways to run the script:

METHOD #1: From Shorthand's Shortcut Menu

1. Place the sh_global_search.tcl file in your Shorthand folder.
2. From Shorthand's **Shortcut** menu, choose **Edit**.
3. Click the **Add** button.
4. Click the **Browse** button and select sh_global_search.tcl.
5. In the Specify a File Shortcut box, specify a keyboard shortcut (e.g. **Shift+Ctrl+F**)
6. In the Description box, type in a short description (e.g. "Search/Replace").
7. Click OK as necessary to return back to Shorthand main window.
8. You can now run the script by invoking the keyboard shortcut or by choosing "Search/Replace" from Shorthand's **Shortcut** menu.

METHOD #2: From Shorthand's File Menu

1. From Shorthand's **File** menu, choose **Tcl Script**.
2. Click the **Open** button and load in sh_global_search.tcl.
3. Click the **Eval Script** button.

How it works

The script determines the dictionaries listed in the Dictionary tabs by reading the PAGES.CFG file located in your Shorthand folder. (Shorthand stores the dictionary tab names in PAGES.CFG.) Since the PAGES.CFG is updated only after Shorthand exits, the script makes sure the active dictionary is always included in the list. The script then uses the `sh_input list` command to present the dictionary files in a pick list in which the user can select which dictionary to search through. The search and replacement strings are obtained from the user through `sh_input string` commands.

The actual search and replace makes use of Tcl's powerful `regsub` command. To be able to search/replace for the special Tcl characters (e.g. braces, double quotes, brackets, backslashes), the script first converts the strings to their hexadecimal values before applying the `regsub` command on the strings.

You can get more information on the `regsub` command from any Tcl book or from the Tcl documentation at http://www.scriptics.com.

Appendix A: Common Problems

Symptom	***Remedy***
Shorthand is not expanding abbreviations.	• Make sure AutoReplace is enabled (p. 51). • Shorthand may be waiting for input; make sure there is no Shorthand dialog box visible (click on the HIDE button if the Main Window is visible). • Make sure the abbreviation is in the active Shorthand dictionary. • Try using Shorthand with NOTEPAD; if Shorthand works with NOTEPAD then Shorthand is working properly and a function in your word processor (e.g. spell checking) may be interfering with Shorthand. • Another application may be conflicting with Shorthand. Disable all programs that monitor keystrokes such as spell checkers, clipboard utilities, mouse utilities (e.g. Power Toys) and built-in word expanders like QuickCorrect in WordPerfect and AutoCorrect in Microsoft Word. • Your computer may have a virus; scan your system for viruses.
Shorthand is expanding text in spurts.	This is by design to give your word processor time to process the keystrokes from Shorthand. You can change the delay (p. 52).
Shorthand is expanding text too slowly.	• Try decreasing the delay between keystrokes (p. 52). • Another application may be interfering with Shorthand. Disable all programs that monitor keystrokes such as spell checkers, clipboard utilities, mouse utilities (e.g. Power Toys) and built-in word expanders such AutoCorrect in Microsoft Word.
Expanded text is missing characters or coming out in parTIAL caps.	Shorthand may be expanding text too fast for your word processor to keep up. Try increasing the delay between keystrokes (p. 52).
Shorthand is expanding inside another word.	This can occur with some wireless keyboards. Try reinstalling your wireless keyboard drivers or switch to a regular keyboard.
After an expansion, all my keys come out in UPPERCASE.	A SHIFT UP key command was lost; reset your keyboard by pressing and releasing the SHIFT key on the left side of your keyboard. Shorthand may be expanding text too fast for your word processor to keep up; try increasing the delay between keystrokes (p. 52).
Simulation of keys in the Numeric Keypad (e.g. {@KEY home} or {@KEY pgup} are not coming out correctly.	Make sure the NUMLOCK key on your keyboard is disabled.
Windows crashes when using Shorthand.	• Run Shorthand last; that is, run your word processor first then run Shorthand. This fixes most memory errors. • Another application may be conflicting with Shorthand. Disable all programs that monitor keystrokes such as spell checkers, clipboard utilities, mouse utilities (e.g. Power Toys) and built-in word expanders like QuickCorrect in WordPerfect and AutoCorrect in Microsoft Word. • Your computer may have a virus; scan your system for viruses.

Appendix B: Choosing the Right Keyword

Choosing the right keywords (abbreviations) can make you work much more efficiently; a good keyword is short, easy to remember and does not conflict with real words.

Create special keywords to prevent unwanted expansions

Shorthand treats all non-alphanumeric characters as word separators so a common problem is unwanted expansions in words with embedded punctuation marks. For example, let's say you have the following Shorthand entry:

```
t = the
```

Now if you type `don't` in your word processor, Shorthand will expand the "t" and you will end up with: don the

To prevent this, define another entry:

```
't = 't
```

The above entry tells Shorthand to expand 't to 't. It may seem strange to define an abbreviation that expands to itself but you can use this trick to prevent unwanted expansions.

An even better solution would be define an abbreviation for "don't" so you don't have to type the apostrophe at all:

```
dont = don't
```

Avoid keywords with embedded punctuation marks

Another common problem is having abbreviations that contain punctuation marks. For example, if you have these 2 entries:

```
t = the
t/o = take out
```

In your word processor, if you type: "t/o", you will get: the/o

To fix this problem, define this additional entry to prevent Shorthand from expanding "t/" to "the/":

```
t/ = t/
```

A better solution would be avoid defining abbreviations with an embedded "/" (or any punctuation marks). For example, you could define an entry for "take out" as:

```
t9o = take out
```

Appendix C: Tips for Touch Typists

Below are some tips on controlling Shorthand entirely through the keyboard:

Use Shorthand's keyboard commands

Shorthand has a keyboard command to access most of its functions. The **F10** (the Hot key, p. 49) brings up the Main Window with Dictionary list box automatically scrolling to the keyword closest to the word you last typed in your word processor. In Shorthand's Main Window, you can press **Ins** to create a new dictionary entry or use your keyboard's **arrow keys** to scroll up and down the dictionary list; you can also type the abbreviation in the Keyword box to quickly jump to that entry. Press **F2** to edit or **Alt+Ins** to duplicate the selected entry in the Dictionary list box. When done, press **Esc** to hide the Main Window and return to your word processor.

Disable the Description box tab stop

It is not necessary to enter something in the Description box when adding a dictionary entry. By default, when you press the **Tab** key in the Text to Type box, your cursor will move to the Description box and you have press Tab again to move to the OK button. You can make the Tab key skip over the Description box and go directly to OK button; here's how to do this:

1. Bring up the Dictionary Text window by modifying any entry (see p. 34).
2. Position your mouse over the Description box and click your right mouse button. A pop-up menu appears.
3. If there is a check mark next to the **Enable Tabstop** option in the pop-up menu, deselect (uncheck) it by clicking on the **Enable Tabstop** option in the pop-up menu.

Use the Alt+Ins key to add new entries

Let’s say you have launched Shorthand (p. 13) and are typing a letter in your word processor. You want to add an entry for “Sincerely yours,” in Shorthand. To do this:

1. Press **Alt+Ins** (p. 55) to create a new entry in Shorthand. Shorthand displays an empty Dictionary Text [Add] dialog box with your text cursor blinking in the Keyword box (p. 27).
2. In the Keyword box, type in your abbreviation (e.g. “sy”) then press the **Tab** key. Your text cursor is now blinking in the Text to Type box (p. 30).
3. In the Text to Type box, type in your long form (e.g. “Sincerely yours,”) and press the **Tab** key. The OK button should be selected (i.e. a dotted rectangle around it).
4. Press **Enter** to save your entry and return to your word processor.

Appendix D: Windows Keyboard Shortcuts

Shorthand can only simulate keystrokes. Fortunately, Windows is designed such that you can access almost any Windows function through a keyboard command. The following table lists some common functions and their keyboard commands:

{@KEY} code	***Function***
`{@KEY alt+tab}`	Activate next window.
`{@KEY tab}`	Moves cursor to the next field in a window.
`{@KEY alt+space}`	Activates the application's main window's system menu (e.g. to minimize or maximize the window).
`{@KEY alt+ –}`	Activates the current window's system menu.
`{@KEY ctrl+esc}`	Activates the Windows START menu.

`{@KEY shift+right}`
`{@KEY shift+left}`
`{@KEY shift+up}`
`{@KEY shift+down}`
`{@KEY shift+home}`
`{@KEY shift+end}`
	Selects text; normally used with Ctrl+C to copy text to the clipboard.

`{@KEY ctrl+c}` `{@KEY ctrl+ins}`	Copies selected text to the clipboard.
`{@KEY ctrl+v}` `{@KEY shift+ins}`	Pastes clipboard text.
`{@KEY home}` `{@KEY end}`	Moves cursor to the beginning or end of the text line.
`{@KEY ctrl+home}` `{@KEY ctrl+end}`	Moves cursor to the beginning or end of the text.
`{@KEY ctrl+left}` `{@KEY ctrl+right}`	Moves cursor one word left or right.

Notes

- After the {@KEY} command to switch or open a new window or menu, you may need to use the {@PAUSE} tag (p.72) to give time for Windows to redraw the screen.
- Word processors (such as Microsoft Word and WordPerfect) have keyboard shortcuts for all their important functions (e.g. change to bold or superscript). If you know the keyboard command, you can simulate the function with Shorthand's {@KEY} tag (see p. 70 for an example).

Appendix E: How to move Shorthand to another computer

To move your Shorthand files to a new computer:

1. Install Shorthand on the new PC.
2. On your old PC, locate your Shorthand folder (see p. 10).
3. Copy the following files from your Shorthand folder to a floppy disk (you may need more than one floppy disk if your word lists are large):

   ```
   PAGES.CFG
   HISTORY.CFG
   SHORTHND.INI
   SH9.LIC
   SHORTHND.TCL
   SH7SPELL.UDT
   ```

   Also copy all your Shorthand dictionary (*.SPF) files to the floppy disk. Note: if you saved your word list (*.SPF) files to a different folder, you will need to locate and copy those .SPF files to the floppy disk also.
4. Make sure Shorthand is not running on your new PC.
5. Insert the floppy disk you made in step 3 to your new PC.
6. Copy the contents of the floppy disk to your Shorthand folder (see p. 10) on the new PC.
7. Start Shorthand on your new PC. Shorthand should pick up your saved settings. However, if you moved your word lists to a folder on your new PC that has a different name or hard drive letter than what was on your old PC, Shorthand may display a message saying it was unable to find a file. If this is the case, you can reopen the file by choosing Open from the File menu.

It is recommended that you DO NOT DELETE any of your files from your old PC until you are certain that you moved all your files and Shorthand works properly on your new PC.

Your Shorthand folder can be determined from the Preferences window's title bar.

Bring up the Preferences window by choosing Preferences *from Shorthand's* File *menu. The title bar of the Preferences window shows the path to the SHORTHND.INI file where your preferences are saved. The folder where you find the SHORTHND.INI file is your Shorthand folder.*

Index

Quick Guide to Common Questions

- **How do I start Shorthand?** See p. 13.
- **How do I know if Shorthand is running?** The Shorthand icon appears in the Windows System Tray when Shorthand is running (p. 13).
- **How do I simulate special fonts such as bold and superscripts?** Use Shorthand's @KEY tag to simulate your word processor's keyboard command to switch fonts (p. 70).
- **How do I make Shorthand pause to accept input?** Use Shorthand's @INPUT tag to get input from the user (p. 76).
- **How do I move Shorthand to a new computer?** See p. 133.
- **How do I create a new, empty word list?** Choose File ⇨ New (p. 18).
- **How do I remove a dictionary tab from Shorthand?** Choose File ⇨ Close (p. 45).
- **How do I merge two or more separate dictionaries into one big one?** You merge dictionaries by *linking* them together (p. 40).
- **Where is my Shorthand USER ID and LICENSE ID?** Choose Help ⇨ About (p. 47).
- **Where is the Shorthand folder?** The Shorthand folder is where the SH9.EXE program is located (p. 133).
- **Why doesn't the text from Shorthand come out in uppercase even though the CAPS LOCK key is on?** CAPS LOCK affects expanded text only if you set the Text Transfer method to "Simulate Keystrokes" (p.54).
- **After an expansion, why are the characters I type coming out in UPPERCASE even though I don't press the SHIFT key?** Your word processor dropped a SHIFT UP command from Shorthand (see p. 129 row 4). Fix this by slowing down Shorthand (p. 52). A setting of Pause 20 msec every 1 keystrokes should work with most computers.