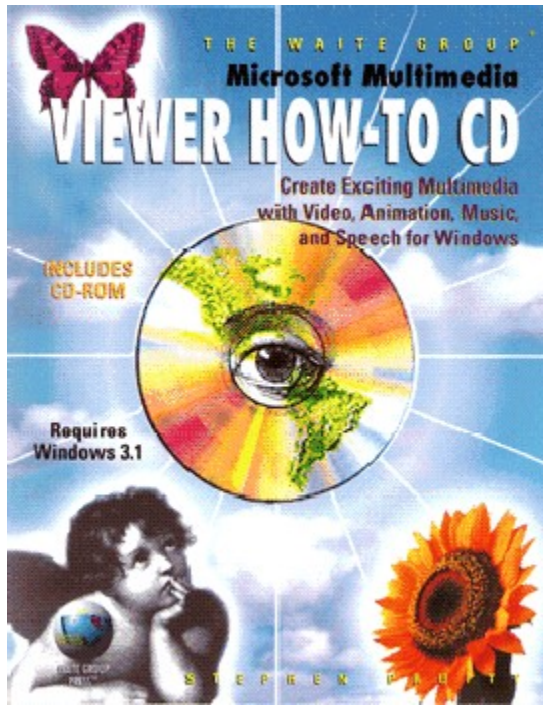


**WELCOME TO THE Viewer How-To!** {ewc MVMCI2,  
ViewerMCI, [device Sequencer][autostart][looping][noframe]intro.mid}



**Please read through this introduction the first time.**

If you've used this application before, you may click on the Contents or Search button above to use this application. Click on the Continue button below to see the introduction.

```
{ewc vwrht2, TsTextButton, "Continue"[Macro=PaneID(qchPath,  
`Intro_2>IntroTxt',55)] [Font="Arial" /S12/B5] /W100 /H40/B1/D2}
```

To learn how to use a Viewer application, choose Viewer Help from the Help menu. This runs a separate instance of Viewer—exit it as though it was running alone.

There are no hot spots outside the table of contents.

```
{ewc vwrht2, TsTextButton, "Continue"[Macro=PaneID(qchPath,  
`Intro_3>IntroTxt',55)] [Font="Arial" /S12/B5] /W100 /H40/B1/D2}
```

The buttons embedded in the text, with labels such as ‘Figure 1–1’, display the indicated figure in a secondary window. The window will always be positioned at the bottom right corner. These buttons cannot be pressed through the keyboard.

```
{ewc vwrht2, TsTextButton, "Continue"[Macro=PaneID(qchPath,  
`Intro_4>IntroTxt',55)] [Font="Arial" /S12/B5] /W100 /H40/B1/D2}
```

The 'Run Demo' button is activated while you are looking at a How-To section. It runs the Viewer application that is created by the section you're looking at. The demo runs a separate instance of Viewer — exit it as though it was running alone.

```
{ewc vwrht2, TsTextButton, "Continue"[Macro=PaneID(qchPath,  
'Intro_5>IntroTxt',55)] [Font="Arial" /S12/B5] /W100 /H40/B1/D2}
```

Be sure to check out the Search operation — especially Searching by Categories. This will make a great reference tool!

The Troubleshooting search category will help you solve some of the problems you encounter as you use Viewer in the future.

```
{ewc vwrht2, TsTextButton, "Continue"[Macro=PaneID(qchPath,  
`Intro_6>IntroTxt',55)] [Font="Arial" /S12/B5] /W100 /H40/B1/D2}
```

The timed appearance of the book cover and this introduction, the buttons embedded in the text, and the custom About box are created using the TSTools addon. The windows containing figures are positioned by a DLL written for this application. No other external functions are used in this application — everything else you see is done with standard Viewer commands.

**Click below to review or click on the Contents or Search button above to begin using this application.**

```
{ewc vwrht2, TsTextButton, "Review"[Macro=PaneID(qchPath,  
`Intro_2>IntroTxt',55)] [Font="Arial" /S12/B5] /W100 /H40/B1/D2}
```



## Table of Contents



Click on a chapter to see that chapter's contents  
(Clicking on Appendix B, C, or D displays that topic directly)  
Click on the page edges to browse through the table of contents

## Chapter 1—What is Viewer?



Click on an entry to display that section

Click on the page edges to browse through the table of contents

## Chapter 2—Designing a Viewer Application



Click on an entry to display that section

Click on the page edges to browse through the table of contents

## Chapter 3—Creating a Simple Application



Click on an entry to display that section

Click on the page edges to browse through the table of contents

## Chapter 4—Adding Graphics



Click on an entry to display that section

Click on the page edges to browse through the table of contents

## Chapter 5—Adding Form and Function



Click on an entry to display that section

Click on the page edges to browse through the table of contents

## Chapter 6—Designing a User Interface



Click on an entry to display that section

Click on the page edges to browse through the table of contents

## Chapter 7—Using Multiple Files



Click on an entry to display that section

Click on the page edges to browse through the table of contents



## Chapter 8—Adding Sound to Your Application



Click on an entry to display that section

Click on the page edges to browse through the table of contents

## Chapter 9—Adding Animation and Movies



Click on an entry to display that section

Click on the page edges to browse through the table of contents

## Chapter 10—Programming with Viewer



Click on an entry to display that section

Click on the page edges to browse through the table of contents

## Chapter 11—Searching for Information



Click on an entry to display that section

Click on the page edges to browse through the table of contents

## Chapter 12—Putting It All Together



Click on an entry to display that section

Click on the page edges to browse through the table of contents

## Appendix A—Installing Your Application



Click on an entry to display that section

Click on the page edges to browse through the table of contents

There are no more pages in this direction

This is a dummy topic that is never displayed. It has no context string or title. It contains a list of the terms used in aliases, defined in the proper fields, so they will appear in the pull-down lists.

```
{vfld137438953482}maximum file size{vfld2850492484943872}  
{vfld137438953482}maximum topics{vfld2850492484943872}  
{vfld137438953482}maximum topic groups{vfld2850492484943872}  
{vfld137438953482}RTF files{vfld2850492484943872}  
{vfld137438953482}Bullet{vfld2850492484943872}  
{vfld137438953482}Em dash{vfld2850492484943872}  
{vfld137438953482}En dash{vfld3413442438365184}  
{vfld137438953484}MVB file not found{vfld3413442438365184}  
{vfld137438953484}search order{vfld280933810831360}
```





Viewer is Microsoft's multimedia authoring system. It is similar to the familiar Windows Help system, with many enhancements. It is designed primarily for electronic publishing tasks, but can do far more.

Viewer includes all of the WinHelp functional capabilities and adds many powerful features, including opportunities for customizing the application's appearance and operation. Viewer can readily be used to provide context help for a Windows application. WinHelp source files can be compiled by Viewer with very few changes.

Viewer makes it easy to create applications. Most of the information is entered by using Word for Windows in conjunction with Viewer programs that let you conduct technical operations, such as multimedia actions, through simple dialog boxes.

While primarily designed for electronic publishing, Viewer can also be used to develop computer-based training and many other types of applications. It is especially suitable for references such as encyclopedias or the popular applications that concentrate on a group of animals such as mammals, birds, or dinosaurs. Viewer can also readily be used to produce more advanced multimedia applications similar to Multimedia Beethoven. This application teaches Beethoven's Ninth Symphony by letting the user play segments of the music while instructive text is displayed. It was one of the first high-quality multimedia applications produced for Windows. Microsoft used Viewer to produce its Bookshelf, Cinemania, and Encarta applications. Viewer was also used to produce the Microsoft Developer Network CDs, which distribute over 100,000 pages of technical information to developers.

Viewer is a cross-platform system. The Viewer Toolkit can create applications that run under Windows, {vfld137438953482}Modular Windows{vfld71919613918576640}, the Sony Multimedia CD-ROM Player, and the Tandy Video Information System. Another company has developed programs that extend this compatibility to include Macintosh systems (Appendix D contains contact information). Viewer's Windows runtime programs can be distributed without paying any royalties.

Version 2.0 of Viewer is a major revision of the original version. The new features include an easy Windows-based authoring system, completely configurable menus and buttons, window panes, additional options in the search function, and a powerful multimedia command.

Viewer is designed especially to create applications that will be distributed on a CD-ROM disk, while providing fast access to the material on the disk.

## 1.1 What Can Viewer Do?

If you have used Windows Help, you've seen some of the basic concepts that are used in Viewer. Viewer builds on these concepts to provide tools that can produce applications that go far beyond simple Help files. You select which of these features are available to the user within a particular application, and control many aspects of the appearance and operation of the selected features.

Viewer applications can present the user with a combination of text, pictures, sounds, animations, and movies. Viewer applications offer the following:

- ü Functions that make sophisticated multimedia events easy to use for both you and the user
- ü An extremely powerful full-text search capability that you can easily customize
- ü A user interface that you can design, incorporating standard or custom menus, buttons, and graphic elements that execute commands when clicked
- ü A wide variety of display options, including master and secondary windows, panes (sections of windows), nonscrolling regions, and popup windows. You can control the size and position of display components or they can be determined automatically by Viewer
- ü Many different ways to navigate through the material in the application, including text and graphic hot spots, buttons that allow browsing through related application sections, a history list that permits returning to previous sections, and a keyword index
- ü The ability for users to print material from the application, or copy it directly to other programs
- ü The ability for users to add personal notes and comments to sections within the application

## 9079244204395200512}

A Viewer application is composed of a series of `{vfld2305856753109041162}topics{vfld2305847957016018944}`. A topic is the basic building block of a Viewer application. All of the material in a Viewer application, including the `{vfld137438953482}table of contents{vfld7278097931641552896}`, menus, and user instructions, is organized into topics. A topic can contain any combination of text, pictures, animations, or movies. If a topic is too big to display all at once, scroll bars are automatically added to let the user see the rest of the material.

Viewer is a `{vfld2305856753109041162}hypertext{vfld-9079251900976594944}` system. This means that you can refer to related material in a way that lets users jump to the related topic by clicking on special text. Users don't have to be forced to follow rigid paths between topics based on menus, but can wander from topic to topic following their interests. You can provide as much, or as little, of this freedom as appropriate or desired.

References to other topics are known as `{vfld2305856753109041162}hot spots{vfld-9079251900976594944}`. A hot spot is a word, phrase, or picture that you marked in a special way. When the user clicks on a hot spot, a specified different topic is displayed, or a Viewer command is executed to perform some other action you defined.

The general process of going from topic to topic is known as `{vfld2305856753109041162}navigation{vfld-9079251900976594944}`. Because you may allow the user to follow an unstructured path through the application's material, you must help the user avoid getting lost. One way to do this is to provide consistent topic headings that help the user understand where the current topic fits into the overall application. Viewer also provides the user with a list that shows the titles of the last 40 topics displayed. The user can return to any of these topics by double-clicking on the desired entry.

### Locating Information

As an author, you can give the user four ways to locate information:

- ü You can define hot spots that form a system of `{vfld2305856753109041162}menus{vfld2305847957016018944}`. These are much like the table of contents in a book or a list of subjects in an encyclopedia. The entries defined as `{vfld137438953482}hot spots{vfld2674570624499712}` display the selected topic when clicked. This works well if the list of topics isn't too long, and if the user can readily identify which topic contains the desired material. Hot spots can also be defined within topics that refer to related topics
- ü You can combine related topics into `{vfld2305856753109041162}browse groups{vfld-`

9079251900976594944}. The user can jump between topics within a browse group by clicking on a button. For example, the Viewer application version of this book is organized with a topic for each section of each chapter. All of the sections within each chapter are organized into a browse group.

- ü You can mark selected terms as `{vfld2305856753109041162}keywords{vfld-9079251900976594944}`. The user can then display a list of these keywords, and easily display any topic that contains a selected keyword.
- ü The user can `{vfld2305856753109041162}search{vfld2305847957016018944}` the entire text of the application for any desired word or phrase, and get a list of topics containing that term within seconds. The topics can be displayed with a mouse click. You can customize the search function to provide `{vfld137438953482}categories{vfld843883764252672}` of terms—for example, to distinguish between Washington State and George Washington. The user can limit the search to groups of topics you define.

### Displaying Information

Viewer applications are displayed in a `{vfld137438953482}main window{vfld11132555231232}`, and can also use one `{vfld137438953482}secondary window{vfld-7997830529422458880}` at a time. You can set the size and position of both windows, and the user can adjust the size and position.

You can divide each window into multiple `{vfld2305856753109041162}panes{vfld2305847957016018944}` if desired. There can only be one `{vfld137438953482}master pane{vfld2850492484943872}`, which is where topics are normally displayed. You can cause topics to be displayed in any other pane, and can control the size and position of each pane. The user cannot adjust panes. Panes are useful for information that should remain visible along with the current topic, such as user interface controls, menus, or pictures.

You can define a `{vfld137438953482}non-scrolling region{vfld7956171150537523200}` in topics that are displayed in the master pane. This stays displayed as the user scrolls through the remaining material in that topic. This is useful for headings or user interface controls.

A hot spot can also cause a small window, known as a `{vfld2305856753109041162}popup{vfld-9079251900976594944}`, to be displayed on top of the normal window. A popup window can contain text or pictures, and disappears when the user clicks anywhere

outside the popup. Popup windows are usually used to display definitions or explanations, but can serve many purposes. Popup windows can even contain other hot spots.

## **{vfld2305854554085785610}Multimedia{vfld-9079244204395200512} and Pictures**

Viewer makes it easy to include multimedia excitement in your application. You can choose between two types of sound files—{vfld137438953482}MIDI{vfld1113255231232} files that play music from a series of recorded notes or {vfld137438953482}wave{vfld-7997830529422458880} files that contain digital recordings of speech, sound effects, music, or any other sounds. You can play movie files that are created from videocameras or videotapes, or animations that are created on a computer.

Viewer makes it easy to control playing a multimedia file. You can play the file automatically, or let the user start, pause, or stop the file. A simple dialog box lets you choose a multimedia file and set some basic options. Buttons bring up other dialog boxes that create a caption or define buttons that let the user control playing the sound or movie.

Viewer makes it equally easy to make your pictures look their best. You can use {vfld137438953482}256-color{vfld1113255231232} pictures, and control how they'll appear on {vfld137438953482}16-color{vfld1113255231232} systems. Viewer gives you an option that handles this conversion automatically, with great results. You can use pictures that are created in one video resolution, such as {vfld137438953482}VGA{vfld1113255231232}, and be confident they will display properly on other systems such as {vfld137438953482}SVGA{vfld7886646831289991168}—without doing anything special!

### **Limits**

Viewer is designed to create large applications—in fact, these applications are expected to be distributed on CD-ROM rather than diskettes. The following limits will help you appreciate how large and complex a Viewer application can be:

- ü A Viewer application file can be as large as 2 gigabytesüThe number of topics in an application is limited only by disk space
- ü512,000 topics can be indexed if you use topic groupsüEach RTF file can contain 32,767 topics
- ü10,000 files can be stored in BaggageüThe stop file can contain 1,024 words to be excluded from the full-text indexüUp to 4 billion aliases can be used

! {vfld137438953482} {vfld2674570624499712}

- {vfld137438953482} {vfld2674570624499712}
- It's easy to control playing a multimedia file. You can play the file automatically, or let the user start, pause, or stop the file. A simple dialog box lets you choose a multimedia file and set some basic options. Buttons bring up other dialog boxes that create a caption or define buttons that let the user control playing the sound or movie. Viewer makes it equally easy to make your pictures look their best. You can use 256-color pictures, and control how they'll appear on 16-color systems. Viewer gives you an option that handles this conversion automatically, with great results. You can use pictures that are created in one video resolution, such as
- Each topic can contain a total of 256 {vfld137438953482} non-scrolling regions {vfld11132555231232}, scrolling regions, {vfld137438953482} panes {vfld11132555231232}, and {vfld137438953482} popups {vfld2674570624499712}
  - Each application can contain 200 {vfld2345052143626} topic groups and word wheels {vfld11132555231232} Each application can have 255 {vfld137438953482} keywords {vfld2674570624499712}
  - Each application can have 255 {vfld137438953482} windows {vfld2674570624499712}
  - Each application can have 255 {vfld137438953482} panes {vfld2674570624499712}
  - Each application can have 255 {vfld137438953482} custom popups {vfld2674570624499712} {vfld137438953482} {vfld280933810831360}
  - Project Editor only supports a total of 153 windows, panes, and popups. Any additional definitions must be manually added to the MVP file

If your application can't fit within these limits, it won't be able to run on anything but very specialized computers.

## 1.2 How Do I Use Viewer?

There are two basic parts to Viewer—an *authoring* system that you use to create a Viewer application, and a *runtime* system that runs the application. You distribute the runtime programs with your application. Chapter 2 describes the authoring system and its use in more detail.

Follow a few general steps to create a Viewer application:

- ü Design the application function and appearance—decide how you want to present information; select the types of pictures, sounds, and movies you will use; decide how users will locate information and navigate between topics; and select the user interface.
- ü Design the details—divide the material into individual displays, select groups of topics and design menus.
- ü Prepare the supporting files—purchase or create the picture, sound, and movie files you will use. This includes obtaining the right to use files you didn't create. You may use temporary placeholders as substitutes for some files, so you aren't held up while you locate all the files you need.
- ü Create the document file—create your text and the commands needed to display pictures, play sounds or movies, and perform other desired actions. This file is the heart of your application. It is created using Microsoft's Word for Windows. One of the programs in the Viewer authoring system, the Topic Editor, makes it easy to create nearly all of the special entries that control your application. It lets you enter most of the necessary information through easily understood dialog boxes.
- ü Compile the files—execute the Viewer compiler to create a Viewer application file from your document and supporting files. This also creates a log file that lists any errors that were found.
- ü Test—run the application, and see if it appears as you intended.
- ü Distribute—prepare your files for distribution, including creating the program that will install the files on the users' systems.



### 1.3 How Can I Make Viewer Do Even More?

Viewer is designed to be extended through external programs in several ways:

- ü Functions in external libraries (DLLs) can be defined and executed within Viewer just like internal commands. This includes Windows APIs (such as MessageBox and sndPlaySound), commercial DLLs (such as the addons on the enclosed CD), and custom-written functions unique to your application.
- ü The embedded window interface is well documented, which enables you to use this technique to develop additional capabilities. A sample program is provided with Viewer, to help you understand the techniques involved.
- ü The standard search operation uses internal functions that are well documented, so that you can develop customized search dialog boxes and operations. A sample program is provided with Viewer to help you understand the techniques involved.
- ü An API function is provided that allows external programs to start a Viewer session and issue commands to control that session.
- ü A programming interface is provided that allows a program to be informed of actions taken by a Viewer application, such as changing the size of a window, displaying a new topic, or scrolling within a topic.

Most of these extension capabilities are used in the “How-To” demonstrations in this book. They are also used by the add-on software that is on the included CD-ROM disk, and described in Appendix B.

## 1.4 Examples of Viewer Applications

One of the best ways to understand the capabilities of Viewer is to examine some Viewer applications. Some of the applications described in this section are available in local software stores or through mail order distributors; others must be ordered from the developer; and three are included on the enclosed CD-ROM disk.

ü

*{vfld2305847957016018954}*Bookshelf*{vfld1406833717673984}* '93 is produced by Microsoft. It is a multimedia reference library that includes an encyclopedia, a dictionary, a thesaurus, an almanac, an atlas, and two books of quotations. The seven references have a combined total of more than 150,000 entries. Readers can watch animations and listen to narrations, quotations, and pronunciations. Entries can be easily copied into other Windows applications such as word processors or spreadsheets. This package can be purchased through software retail stores or directly from Microsoft.

ü

*{vfld2305847957016018954}*Encarta*{vfld-9223358292959428608}* is a multimedia version of the 29-volume 1992 Funk & Wagnalls New Encyclopedia, plus more than 1,000 original articles. It is produced by Microsoft. It includes animations, thousands of photos, and hours of music and speech. There are more than 25,000 articles and 17,000 multimedia elements. Encarta includes several unique features in its user interface, including Timeline, Category Browser, and Research Wizard. This package can be purchased through software retail stores or directly from Microsoft.

ü

*{vfld2305847957016018954}*Cinemanía*{vfld2674570624499712}* is an interactive movie guide produced by Microsoft. It lets you see capsule summaries, reviews, photos, definitions of terms, and film histories, and hear 100 of the most famous lines. Users can locate movies based on genre, actor, director, release date, star rating, Academy Award, and MPAA rating. This package can be purchased through software retail stores or directly from Microsoft.

ü

The Microsoft *{vfld137438953482}*Developer Network*{vfld9288133065572352}* CD is issued quarterly to distribute more than 100,000 pages of technical information to developers—including reference books from all developer-related

software, the database of known bugs, workarounds and technical tips, sample programs, technical articles, the complete text of several years' issues of the MS Systems Journal magazine, and the complete text of a commercial book. Users can obtain a list of materials that refer to an area of interest within seconds—and can look at any of the listed articles with a simple click of the mouse. This product can only be obtained directly from Microsoft by calling (800) 759–5474.

ü

{vfld2305847957016018954}Gallery{vfld2674570624499712} is a Viewer demonstration package created by Microsoft. It is included on the enclosed CD–ROM disk. Gallery uses graphics for the user interface, and demonstrates many of the abilities of Viewer. An icon to run Gallery is created on your system when you run the Viewer installation.

ü {vfld2305847957016018954}USA Tour{vfld280933810831360} is another demonstration package created by Microsoft and distributed with Viewer. It provides an outstanding demonstration of a custom user interface. It can be found on the enclosed CD–ROM disk as \MVSAMPLE\USA\USA.MVB.

ü The {vfld137438953482}Gateway Mall{vfld-9223091103043944448} is a multimedia buyers guide showing part of Gateway 2000's product line. It uses sounds, animation, and high–quality pictures to describe features of items such as tape drives, notebook computers, and video boards. Figure 1–1 shows the introductory window.

```
{ewc vwrht2, TsTextButton, "Figure  
1ü½1"[Macro=JI('viewerht.mvb>SecWin', `fig1_1')]  
[Font="Arial" /S12/B4] /W100 /H40/B1/D2}
```

Figure 1–2 shows an example of product information. The Gateway Mall CD is included with new Gateway 2000 systems that include CD drives. Future versions may be available through other means as well.

```
{ewc vwrht2, TsTextButton, "Figure  
1ü½2"[Macro=JI('viewerht.mvb>SecWin', `fig1_2')]  
[Font="Arial" /S12/B4] /W100 /H40/B1/D2}
```

ü The {vfld137438953482}Merck Manual{vfld-

9223091103043944448} is a comprehensive database of medical information for doctors, pharmacists, and other medical professionals. It was developed by Keyboard Publishing, which also provided many of the Viewer extension software packages included with this book. Figure 1–3 shows the results of a search operation in this package.

```
{ewc vwrht2, TsTextButton, "Figure  
1i½3"[Macro=JI('viewerht.mvb>SecWin', `fig1_3')]  
[Font="Arial" /S12/B4] /W100 /H40/B1/D2}
```

ü The {vfld137438953482}Hugo and Nebula Anthology {vfld-7998112004399169536} 1993, produced by ClariNet Communications, contains the complete text of all the 1993 nominees for the top science fiction awards. It includes five novels, nine novellas, nine novelettes, many short stories, nominees for art awards, and a mass of other materials too extensive to describe. Figure 1–4 shows the opening screen and a custom menu that reflects the unique composition of this work.

```
{ewc vwrht2, TsTextButton, "Figure  
1i½4"[Macro=JI('viewerht.mvb>SecWin', `fig1_4')]  
[Font="Arial" /S12/B4] /W100 /H40/B1/D2}
```

Figure 1–5 shows one of the menus, which lets the user click on the spine of the desired book. This CD can be purchased from Clarinet by calling (800) ORDER–BOOKS.

```
{ewc vwrht2, TsTextButton, "Figure  
1i½5"[Macro=JI('viewerht.mvb>SecWin', `fig1_5')]  
[Font="Arial" /S12/B4] /W100 /H40/B1/D2}
```

ü Microsoft Multimedia How–To has also been prepared as an “electronic book” using Viewer. It lets you search for subjects, commands, or other terms used in the book, run each of the How–To demonstrations, and print sections so you can write on the steps as you follow them. Figures are displayed in a secondary window when you click on the appropriate button—this lets you move and resize them alongside the text if you want. Search categories let you locate references to internal and external commands,

Viewer terms, and other material. The electronic book is designed to serve as a reference tool as you develop your own applications. An icon to run this application is created on your system when you run the Viewer How-To installation. Figure 1-6 shows a typical screen in this application.

```
{ewc vwrht2, TsTextButton, "Figure  
1-6"}[Macro=JI(`viewerht.mvb>SecWin', `fig1_6')]  
[Font="Arial" /S12/B4] /W100 /H40/B1/D2}
```



CONTINUE...

DETAILS

PRICE INFO

HOW TO ORDER



Product Details



The Gateway 2000 Nomad

Gateway 2000™ Nomad  
power. They boost up  
with battery life that lit

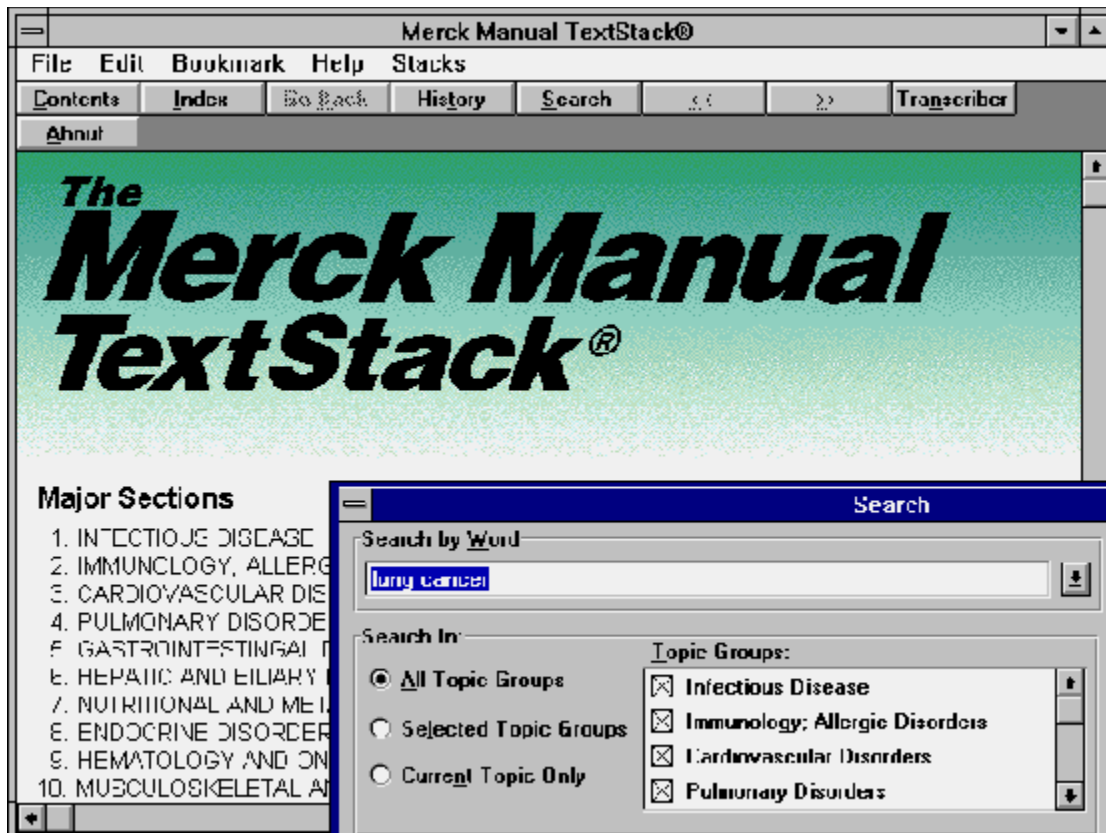
The Nomad's 13-inch  
come with a portable  
parallel, external keyb  
is supplied by a power  
or an AC adapter. Both  
touch trackball, and esc

Nomad 4255XL

Intel 25MHz 486SX processor, 120MB IDE hard drive, 4MB RAM (expandable to 20MB),  
60ns static column, 1MB video memory

Nomad 4500XL 120

Intel 50MHz 486DX2 processor, 120MB IDE hard drive, 4MB RAM (expandable to 20MB),





Hugo and Nebula Award Anthology 1993

File Edit Bookmark Help Fiction Art Other

Contents Go Back History Search Hugo Novels

- Hugo Novellas
- Hugo Novelettes
- Hugo Short Stories


---

- Nebula Novellas
- Nebula Novelettes
- Nebula Short Stories

---

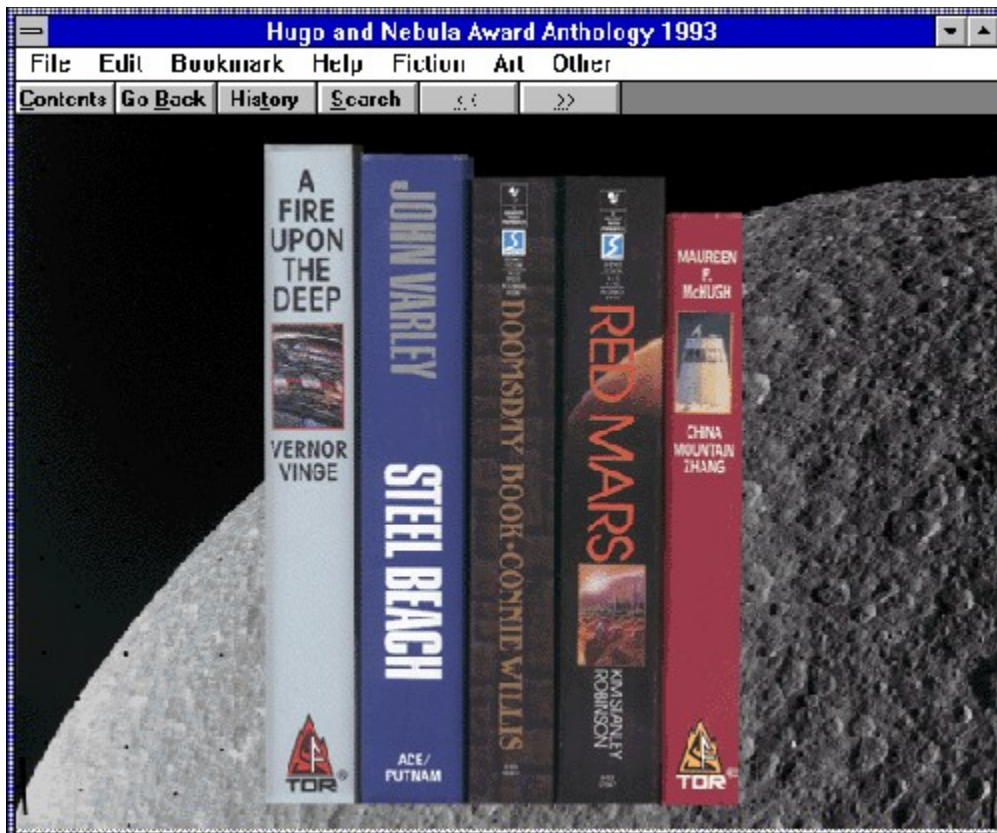
- Campbell Nominces

*Hugo and Nebula Anthology 1993*



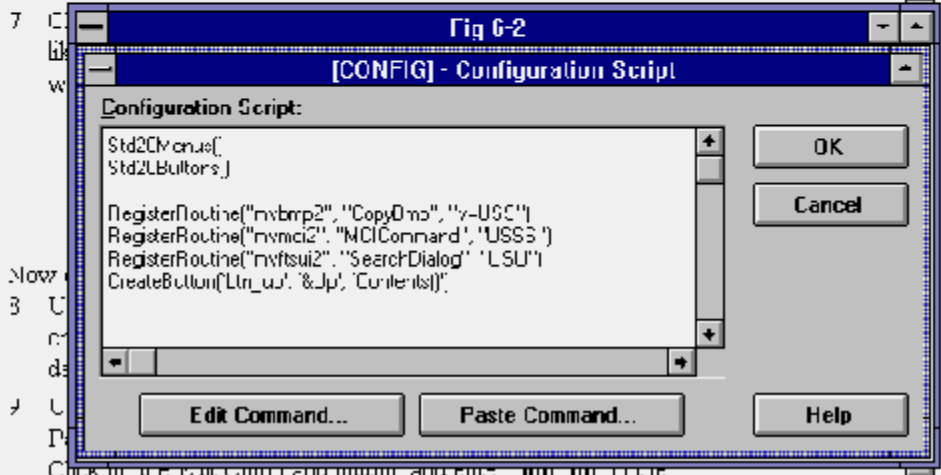
CLICK ON BOOKS TO CONTINUE...

ClariNet  
Communications Corp



then select the Contents() command. The dialog box should look like Figure 6-1.

FIGURE 6-1





Creating a multimedia application using the Microsoft Multimedia Viewer package is very simple and straightforward. In this chapter you learn how to create an application and are introduced to Viewer's files and tools. You also learn how to approach application design, and how to select the pictures, sounds, and movies to include. Finally, you discover how to get more advice and assistance if you have questions or problems while working with Viewer.

The first and most critical step is designing your application. Start by deciding how to present information to your users and what types of pictures, sounds, and movies to offer. Then decide how your users will locate the information they want and how they will navigate between sections of your application. Finally, decide on an interface that makes it easy for the users to see and hear your material, locate specific information, and navigate from place to place in your application.

After you complete your high-level design, it's time to design the details. Decide how to divide the material into individual displays, how to group these displays, and all the other details of implementing your general design.

After you complete your design, prepare the files. This can include purchasing some supporting files such as pictures, sounds, or movies. It also includes obtaining publication rights to other files and creating some of your own. You also prepare one or more document files that contain your text and define the organization of all of your information. The Viewer compiler uses the document and supporting files to produce your application.

As you complete portions of your document files, run the Viewer compiler to combine all of the pieces into a Viewer application file. Review any error messages provided by the compiler to signal problems in your material, and test your compiled file by playing it with Viewer's runtime program.

After your application is complete, prepare to distribute it to your users or customers. This process is explained in appendix A.

## 2.1 How Should I Design the Application?

It is important to understand how a Viewer application is organized so you can understand the design concepts. The basic building blocks of a Viewer application are called `{vfld2305858952132296714}topics{vfld-9079242005371944960}`. A topic is a segment of your application that is displayed as a result of an action by the user. In the simplest cases, these are the places in your application that the user can jump to.

Topics can contain any combination of text, pictures, controls, or other visible parts of your application. Sounds or movies can be played automatically when the topic is displayed, or when the user selects one of the *controls*. Controls are portions of the text or pictures that cause a different topic to be displayed in the master pane of the main window, in a different pane of the main window, or in a different window.

All topics must have an internal name, known as a `{vfld2305858952132296714}context string{vfld-9079242005371944960}`. The context string is used by a Viewer command to reference a specific topic so that the command can display the intended topic. You learn how to create topics in Chapter 3.

This book presents a very general introduction to designing Viewer applications. The great variety of techniques that Viewer supports, combined with the infinite range of possible applications, makes application design too large a subject to cover here in depth. This book is intended to give you an appreciation of the important components of design, and an understanding of the techniques available. After reading this book, examine as many Viewer applications as possible, and see if you can figure out how they produce the effects that you like. Identify how the authors decided to handle each of the design components described in this section. Look at other applications that have something in common with your own, and try to decide how you could produce the same effects in Viewer. Reading this book and studying actual applications will give you many ideas to apply to the design of your own application.

### Decide How to Present Information

The first step in designing your application is deciding how you will present information to your users. There are two aspects to consider: the user interface, which consists of the menus, buttons, or other controls you will present to allow a user to access certain information, and the display of the actual information. These designs may not be fixed across your entire application. You might use several types of controls in different parts of your application, and you might present your information differently in each part. Windows that provide information can also include controls.

You can use several different combinations of features:

- ü The primary portion of a topic display—the focus of interest—can be either text or a picture. Topics that serve the same purpose should ordinarily use the same design. This helps keep the material clear to the user.
- ü More than one topic can be displayed at once within the main window, using a feature known as `{vfld2305858952132296714}panes{vfld-9079242005371944960}`. Only one of these topics can be larger than the display area; scroll bars to let the user display the

remaining material. The other topics are limited to the maximum size defined for the panes where they are displayed.

- ü You can create panes that contain only a set of controls, providing a custom user interface. These controls can change based on what part of the application is being displayed, or they can be fixed throughout. A pane can be located anywhere in the window. There can even be more than one such pane, to place controls in several areas of the window.
- ü You can keep all of your topics small, so that the user never sees a scroll bar. This looks like a series of index cards (similar in effect to the Apple Hypercard system).
- ü Additional topics can also be displayed in separate windows, known as `{vfld2305858952132296714}secondary windows{vfld-9079242005371944960}`. These windows can be moved and resized by the user, just as any other window. They remain open until closed by the user or by an explicit Viewer command, or until the application is closed.

### **Select Types of Pictures, Sounds, and Movies**

The second step is to determine what types of pictures, sounds, and movies you want to present to the user. Viewer can support 16- and 256-color pictures, Wave and MIDI sound files, video movies, and animation files. The next section of this chapter, and the chapters that cover those specific subjects, explain in detail the capabilities of Viewer and the effects your decisions might have on the disk space your application requires.

### **Select Tools for Locating Information and Navigating**

The third step is to decide what tools you will give the users to locate information in your application, and how they will navigate between topics in your application. Viewer places a number of extremely powerful functions at your disposal:

- ü You can define `{vfld2305858952132296714}hot spots{vfld-9079242005371944960}` in your text or pictures to display different topics. These can be used to create text or graphic menus, or can be included within normal topics to provide references to related material.
- ü You can combine selected topics into groups and allow users to browse among the topics in each group.
- ü You can provide the ability to search all of the text in your application for any word that appears, or restrict this search to selected groups of topics.
- ü You can provide one or more lists of `{vfld137438953482}keywords{vfld4363420484763648}`, which you select, that can be used to select desired topics.
- ü You can allow the user to search for topics based on values in `{vfld137438953482}fields{vfld-9006918320930160640}` that you define within the text. These fields can use numeric, date, and era (such as 12,000 B.C.) forms.

### **Design the User Interface**

The last step of the high-level design is to decide on a user interface. The

interface is simply the set of controls available to the user. It can include the following:

- ü Some, none, or all of the standard menus and buttons
- ü Additional custom menu items or buttons
- ü Graphic controls
- ü Graphic or text hot spots within topics that include other material
- ü Custom-written Search dialogs

### **Design Implementation Details**

Now it's time to decide on the details that implement your design. You have to decide what windows and panes will be needed, what material will be in each topic, which topics will be grouped together, and which picture, sound, and movie files you will use. A critical part of this is deciding which topics will be referenced by which other topics. If you don't design these paths between topics in advance, you may create "orphan" topics that are not referenced anywhere else, and cannot be reached by the user.

You should decide on the names of each topic, since you will want to create hot spots and controls before all of the topics have been defined. It would be difficult to go back and add all of the hot spots and controls after all the topics have been defined. It's much easier to select the context strings as you design the application, then define the hot spots as you write each topic. As you decide on these details, you should record them in a document that you can refer to as you create your application. This document should include the title and context string of each topic, the context strings of other topics that this topic references, and which other topics reference this one. This document should be in a file on your computer, so that it's easy to update and to print new copies as necessary. Keep it at your fingertips as you work—it will become the "bible" of your application development.

Decide on a standard way to name the context strings after you decide on your overall structure. That helps assure that the names will be meaningful and helps prevent accidental duplications.



## 2.2 Viewer's Files

A Viewer project can include many different types of files. This section divides the files into three simple categories to make them easier to understand.

- ü First are the required files that you create. These are the files you work with the most. They usually contain the most information, and take the most time and effort to prepare.
- ü Second are the other files used to create an application. These files are necessary, but you usually don't have to do anything with them. They are installed on your system as part of the Viewer package. They provide functions that you can customize if you want, but are usually left unchanged.
- ü Third are optional files that are supported by Viewer. These are primarily the pictures, sounds, and movies that can make an application exciting or provide the user with additional ways to locate sections of your application easily.

This section also explains Viewer's *Baggage* system for storing picture, sound, and movie files, how to estimate the disk space required by an application, and how to reduce the space needed.

### Required Files Created by the Author

You create two primary files to set up an application: the project file, also known as the MVP (Multimedia Viewer Project) file, and the document file, also known as the RTF (Rich Text Format) file.

Each project has one

{vfld137438953482} MVP file {vfld2334552915672301568}. It contains:

- ü Configuration information for the Viewer compiler
- ü Characteristics of the windows and panes you define
- ü A list of Viewer commands to be executed when your compiled file is loaded
- ü Definitions of external functions (DLLs) to be executed by your application
- ü Definitions of fields and data types used to define Search functions
- ü A list of files to be copied into the Baggage file section

The MVP file is created and updated by the Viewer Project Editor program. The Viewer Topic Editor program also updates some information in this file. Use of Project Editor and Topic Editor to control this file is explained throughout this book. The Viewer documentation explains the elements of the file.

The MVP file can be displayed by loading it into Notepad. Examine the MVP file for your projects occasionally to become familiar with the contents and format. This familiarity can be useful if you experience problems you cannot explain, which could be caused by accidental damage to the file due to mistakes using Project Editor or Topic Editor.

Most of the information in your application is in one or more document (RTF) files. A project can have as many document files as desired. Document files contain:

- ü Text to be displayed
- ü Text and picture hot spots
- ü Viewer commands to display pictures and play sounds or movies
- ü Viewer commands to control the user interface
- ü Definitions of context strings with their association to topics
- ü Definitions of keywords to be used for searching for information
- ü Association of topics into topic groups

{vfld137438953482} Document files {vfld8449315309734592512} are created by Microsoft Word for Windows (referred to as “Word” throughout the remainder of this book). The contents of these files are explained in detail throughout this book. You can break a project up into many separate document files, if desired. This can be handy if the file becomes too large to work with effectively; it also allows several authors to work on different parts of a project simultaneously. All command and format information is stored as character–string commands. Documentation describing the format of this file is available, but you should never need to use it, because you will use Word to create this file. You must use Word to be able to use Topic Editor, because Topic Editor is designed specifically to work with Word. You can examine an RTF file by loading it into Notepad, or loading it into Word and changing the File Conversion to Text. Become familiar with the appearance of common portions of an RTF file, such as footnotes and formatting information. If you have problems that you can’t explain, they sometimes can be understood by examining the relevant portion of the RTF file.

Viewer controls features of the application by taking advantage of features of Word that aren’t needed, such as footnotes and selected format options. The features of Word that are used don’t have any logical connection to the associated Viewer features—they are chosen for the sake of convenience. This can be confusing if you use the Word features directly. Fortunately, this isn’t necessary—Topic Editor creates most of these entries for you. Still, it is useful to understand the features and their characteristics so you can use them effectively. Once you are familiar with them, you will find occasions where it is easier to create them directly, without using Topic Editor. Viewer uses six methods to control its features:

- ü {vfld137438953482} Hot spots {vfld4363420484763648} are defined through character formats. These are represented as single or double underlined text immediately followed by hidden text. Topic Editor creates the proper formatted entries.
- ü {vfld137438953482} Non–scrolling regions {vfld11132555231232} and {vfld137438953482} non–wrapping text {vfld12232066859008} are defined through paragraph formats—  
  - {vfld137438953483} Keep with Next {vfld12232066859008} and
  - {vfld137438953483} Keep Together {vfld4363420484763648},
respectively. You must set these formats yourself through Word’s menus.
- ü Definitions that apply to topics, such as context strings, titles, keywords, groups, and topic–entry commands, use special {vfld137438953482} footnotes {vfld3658633531359232}. Instead of Word’s default—identifying footnotes with numbers—these definitions use characters as footnote codes. For example,

titles are defined by footnotes with a dollar sign. Topic Editor creates these footnotes.

- ü Features that must be placed within the text, but are too complex to use special formats, use special sequences of characters known as {vfld137438953482}RTF codes{vfld4363420484763648}. These are based on the codes that are used in the RTF files, but are unique to Viewer. Viewer uses this method to define phrases that are to be used in special ways during text search operations.
- ü {vfld137438953482}Embedded windows{vfld-9006918320930160640} are located within topics. They are used to display pictures, movies, animations, or custom-defined images such as buttons or list boxes. They are defined using RTF codes and specify the name of a program that will be executed to display the requested material. Viewer comes with the programs needed to display 16- or 256-color pictures, movies, and animations. These embedded windows can be defined through Topic Editor dialog boxes. Other embedded windows can be created through Viewer add-on programs, and must be created by typing in the necessary codes. The programs defined are executed as soon as their portion of the topic is displayed, and they are terminated when their portion is no longer displayed.
- ü Viewer also allows specified {vfld137438953482}commands{vfld-7998112004399169536} to be executed under a number of conditions. These commands can cause different topics to be displayed, display topics in specified panes or windows, or change the size or position of windows or panes. They can also execute Viewer functions normally associated with menu items or buttons, such as printing the current topic. External functions can also be executed as Viewer commands. This includes multimedia functions, windows API functions, and custom programs. These commands can be executed when a window is opened (including the main window); when a topic is displayed; or when a hot spot, menu item, or button is clicked. Commands can be defined in each of these events through Topic Editor. In addition, Topic Editor assists in defining commands that are part of Viewer.

The result of your work is a file named *projectname.MVB*, created by the Viewer compiler. The file extension stands for “Multimedia Viewer Book.”

### **Other Files Used to Create an Application**

Viewer uses several standard files that control language-related functions:

- ü The {vfld137438953482}stop-word{vfld-9223356093936173056} file specifies common words to be excluded from the search index, such as *and*, *or* and *the*. The English-language file is USA.STP.
- ü The {vfld137438953482}search-operator{vfld-9223356093936173056} file defines the logical operator terms that can be used in the search dialog box, such as *and*, *or* and *not*. The English-language file is USA.OPR.
- ü The {vfld137438953482}character-

handling table {vfld280933810831360} defines how characters are handled during indexing, such as converting upper case letters to lower case or handling ligature characters such as æ. The English–language file is ANSIUSA.TBL.

These files are supplied with Viewer. They can be modified for your application, although that is rarely required. Standard files are provided for English, French, and German {vfld137438953482} languages {vfld4081945508052992}. The use and format of these files is explained in the Viewer documentation.

The Viewer compiler creates various temporary files that are deleted when the compiler finishes. These files are created in the directory defined by an entry in your computer’s AUTOEXEC.BAT file that looks like SET TEMP=C:\WINDOWS\TEMP. You should have similar SET statements for both {vfld137438953484} TMP {vfld13331578486784} and {vfld137438953484} TEMP {vfld13331578486784} that point to an empty directory. Many Windows programs, such as the Viewer compiler, use one or both of these variables to determine where temporary files should be created. These variables should never refer to your root directory, due to the limited number of files that can be created there, or to a remote device on a {vfld137438953484} LAN {vfld-35184913254711296}, due to the tremendous amount of information that must be written and read.

If you request that your application be compiled with {vfld137438953484} high compression {vfld11132555231232}, the Viewer compiler also creates a {vfld137438953482} phrase table {vfld-9223356093936173056} in a file named *projectname*.PH. This file contains intermediate information used in the compression process. Microsoft recommends that this file be erased between compilations if you make significant changes to the topic content. Reusing a PH file speeds up the compilation greatly if you make few changes. You have to use the Windows File Manager or some similar program to erase this file. Project Editor does not have a function to do this.

The Viewer compiler also creates a file named *projectname*.LOG, containing messages describing any errors that were found during compilation. The file is recreated each time your application is compiled. Project Editor provides an option for displaying this file.

### **Optional File Types Supported by Viewer**

You can create an {vfld137438953482} alias file {vfld-9223356093936173056} to define searchable data that is different from the displayed text. This lets you display terms or numbers in ways that are easy for the users to understand, and use the same information in a different format to allow the user to search for it in a more general form. For example, you might define “7/4/1776” as a searchable alias for the text “Independence Day.” The use of aliases is explained in Chapter 11. This is a standard text file, commonly named ALIAS.TXT or *projectname*.TXT.

You will certainly want to include some optional files, such as pictures, sound files, movies, and animation files. These files make your application interesting and exciting!

Viewer supports several of the common formats for picture files—DIB, BMP, and WMF. It does not support

{vflid137438953484} RIFF DIB {vflid13331578486784} files that can be created by some advanced painting programs. In addition, some programs create {vflid137438953484} BMP {vflid280933810831360} files in nonstandard formats. The Viewer Convert utility and Windows Paintbrush will usually convert such files to an acceptable form. Viewer also accepts a special graphic format known as SHG. These are bitmap files that contain one or more hot spot regions. These files are created by the Viewer Segmented Hotspot Editor program, which is described in more detail in section 2.4.

There are two formats for Windows-supported sound files—{vflid137438953482} Wave {vflid11132555231232} and {vflid137438953482} MIDI {vflid72057052872048640}. Wave files have the file name extension WAV. They are digitized recordings of sounds, including voices, noises, or music, captured through a microphone or other input device. They can be quite large. MIDI files have file name extension MID. They contain commands to music synthesizers, such as those found in sound cards, so they can only play music or other standard sounds supported by synthesizers. The quality of MIDI sound can be excellent, and the files are usually quite small, but the user must have a sound card and driver software installed to be able to play sound files. The use of sound files is explained in detail in Chapter 8.

Viewer also supports any {vflid137438953482} movie {vflid11132555231232} or {vflid137438953482} animation {vflid11132555231232} files that have Windows {vflid137438953482} MCI {vflid1406833717673984} drivers installed. The drivers needed for AVI movies are readily available from Microsoft and can be distributed freely. Special packages of hardware and software are required to create such files. Animation files can be created by packages sold by several vendors. The most popular PC-based animation packages are sold by AutoDesk and Gold Disk. Animation files produced by these packages can be played by Viewer if you have the necessary MCI drivers installed. These drivers would also have to be distributed with your application. The use of movies and animation is explained in more detail in Chapter 9.

### **The Viewer {vflid137438953482} Baggage {vflid-9078975914968088576} File System**

Viewer allows you to incorporate your picture, sound, and movie files right into the compiled MVB file. This is known as storing those files in Viewer's Baggage section. Files are copied into Baggage by the Viewer compiler as a result of listing the file names in the Baggage section of the MVP file. This results in having a single data file installed on the user's system, rather than a large group of files.

The use of a single data file provides one vital benefit to both you and the user—it assures the integrity of your application by making it impossible for files to be erased or replaced, whether by accident or intent.

The {vflid137438953484} Baggage {vflid7882987656592752640} system does not provide a directory structure—all files are referenced by their names, in standard DOS format. To help make up for this, the file names are case sensitive. That is, the files test.wav, TEST.WAV, and teST.wav are completely different and separate. If you want to include files from different

directories with identical names in your application, you can only distinguish them by using different combinations of uppercase and lowercase letters in the file names. The file names must be entered in exactly the same way in the project file and in the command within the document file. Within the document you indicate that a file is to be read from Baggage by preceding the file name with an exclamation point—for example, !test.wav.

Topic Editor provides an option to store the selected file in the Baggage section while defining the commands that use these files. This automatically inserts the file name in the Baggage section of the project file and prefixes the file name in the command with the necessary exclamation point.

It is important to understand that you can't always put all of your files in {vfld137438953484}Baggage{vfld13331578486784}. The program or function being executed by Viewer to use the file must be written especially to understand the exclamation point notation, and have special instructions to read Baggage files. Not all Viewer functions have this capability. For example, Viewer allows you to cause another program to be executed by executing the function {vfld137438953484}ExecProgram{vfld-9223356093936173056}('programname.exe', 0). The program to be executed cannot be stored in Baggage. As a general rule, only the functions unique to Viewer and that use picture, sound, or movie/animation files can utilize Baggage. If Topic Editor does not provide the Baggage option for a command, it's pretty safe to assume that that command can't use it, and that any associated files must remain outside your MVB file.

You can have just some of your files included in Baggage—the use of this feature is strictly optional. This reduces the benefit of Baggage, but it is necessary sometimes.

### **Estimating Disk Space Requirements**

During the development process, three groups of files require space on your system. The first group is the source files—the document, picture, sound, movie, and related files that you produce. The second group is the work files used while your application is being compiled, and the third group is the compiled MVB file. These sets of files do not all have to be located on the same physical drive. In fact, your application may be compiled much faster if the work files are on a separate drive. This can be very important if you are compiling an extremely large application.

There is no fully reliable way to predict the size of your source files for any given application. You have so many choices in the use of text, pictures, and multimedia functions that no two authors are likely to produce similar-sized files for similar applications. These choices are much of what makes Viewer such a powerful authoring system!

The best way to estimate the size of your source files and resulting compiled file is to actually produce a reasonable-sized portion of your application, and develop an estimate from that. You must be sure that the portion created is typical in its use of pictures and multimedia, or allow for any differences in your estimate.

Microsoft has provided a guideline for estimating the size of the temporary work files:

Multiply the number of words in your document file by 128.

Add 2 times the number of topic groups defined.

Add the total number of topics, divided by 4.

Microsoft also suggests that the compiled MVB file will be at least twice the size of your document files, plus the size of your Baggage files.

For example, an application with 15,000 words in 100 topics with 12 topic groups will require

$15,000 \times 128 + 12 \times 2 + 100 / 4 = 1,920,049$  bytes of disk space.

That's over 1.8 megabytes. One author, developing an extremely large application, estimated that the application would have about 10 million words in 100,000 topics. This works out to about 1.3 gigabytes of disk space required for temporary work files!

This large application is expected to have about 50 megabytes of document files and 450 megabytes of Baggage files. This would result in a compiled file of at least 550 megabytes. The combined requirement for source files, work files, and compiled result is over 2 gigabytes!

### **Methods for Reducing the Size of Your MVB File**

The easiest method for reducing the size of your MVB file is provided by Viewer. You can specify, through Project Editor, that you want to `{vfld137438953482}compress{vfld7882987656592752640}` your file when it is compiled. You can choose no compression, medium compression, or high compression. Compression affects only the text portion of your application, not files included in Baggage. Medium compression reduces the size of your file (other than Baggage) by 5–10 percent, and high compression reduces the size by 35–50 percent. The higher degrees of compression cause the time required to compile your application to increase. Most authors do not compress their files until a project is nearly completed, to save time. However, it is a good idea to perform compressed compiles occasionally throughout project development to help estimate the size of the final file and avoid last-minute surprises.

High compression compiles use a phrase table file (*projectname.PH*), which is described earlier in section 2.2. This holds much of the information used to compress the file. The compiler will reuse an existing phrase table if it exists to speed up compilation. It must be erased before compiling if the document text changes significantly, or the compression will be based on obsolete information.

The size of your Baggage files can also be an important part of the size of your MVB file, because they can be very large and they are not compressed by the Viewer compiler. A single 320x200 bitmap with 256 colors requires 64,000 bytes. A wave sound file that plays for only a few seconds can occupy 250K–500K. It is important to estimate the amount of storage that might be needed for such files while you are designing your application. One author discovered that his planned application would require a number of full CD-ROM disks because he planned to include very many large, high-quality pictures. It would be impossible to produce, install, or use such an application! He must reduce the number of pictures, their quality, or both.

Different picture file formats require far different amounts of storage. For example, a 256-color bitmap requires twice the space of the same size 16-color bitmap. Including both formats of a single picture increases the space still further. A bitmap in Super VGA resolution is 1.5 times the size of the same picture in VGA resolution. An image in 8514 resolution is 1.6 times the size of the Super VGA picture! In general, greater degrees of picture quality

require greater amounts of disk space. It is vital that you understand the effect of your choices, so you can make sure the quality is worth the impact. This is especially critical if it affects the distribution of your application. It could increase the number of diskettes needed, or force you to distribute your application on a CD-ROM disk even though you didn't want to limit your market to users with CD drives.

There are a few options for reducing the size of picture files. Shrinking these files can have a significant effect, because most applications will have a large number of images.

One possibility is to use a drawing program, which creates Windows Metafile (WMF) files, instead of a painting program which creates bitmap (BMP) files. WMF files are much smaller than BMPs. Drawing programs have different purposes and features, so they may not be suitable for your purposes. Corel Draw is reported to be the only program able to convert BMPs to WMFs, but it costs far too much for most people to buy just for that purpose.

Another option is to load your BMP files into Viewer's Segmented Hotspot Editor utility (Shed2) and save them as SHG files without defining any hot spots. The resulting files will normally be far smaller than the original BMPs with no loss of quality. They can be used exactly as the BMPs would have been in all Viewer commands. Many authors use this technique.

A third option is to use the Convert utility to convert BMP files into DIB RLEs. These take even less space than SHGs, without any loss of quality. RLE files also load faster and use less memory. Convert can process batches of files together, which makes it easier to use than Shed2 for this purpose.

The last resort is to reduce the quality of the picture by reducing the resolution or number of colors. BitEdit can convert 256-color pictures to dithered 16-color pictures with a minimal loss of quality, while cutting the size of the file in half. The resolution can be reduced in many painting programs by loading and saving the file while running Windows in the desired reduced resolution.

Similar format issues apply to sound files. If you want to play music, keep in mind that a MIDI file requires *much* less space than a Wave file that plays for the same length of time. Of course, a MIDI file won't help if you want to play recorded voices. Wave files can be recorded with different sampling rates, which serves as the audible equivalent to changing the number of colors in a picture—you can reduce the file size at the expense of reduced quality. The type of sounds being recorded, and their purpose, can make all the difference in such choices. Human spoken voices, for example, rarely need high sampling rates. Most listeners would not detect any difference if lower rates were used, which can make longer recordings practical. The opposite extreme would be recordings of samples by symphony orchestras to be heard by serious music buffs. This type of listener might be able to detect the slightest imperfections in your recording. They would also be likely to have the high-quality sound boards and faster computers needed to play files created with high sampling rates accurately.

When considering movie and animation files, remember that the file size is affected by the size and quality of the image and the number of frames per second. Animation files, which are drawn on the computer, tend to require far less space than true video movies—but you must sacrifice some detail in the picture. The size of video movies can be reduced by using advanced



compression techniques, but these require special boards on the PCs used by you and all users of the application. This is impractical unless you are developing an application that will only be used within your company.

## 2.3 What Are the Authoring Tools?

Viewer includes two programs to simplify the authoring process—Project Editor and Topic Editor. These programs make it practical for authors who are not technical experts to create powerful and sophisticated Windows multimedia applications. They simplify the technical details while permitting you to manage these details to the degree desired.

### The Viewer Project Editor

Project Editor, as its name implies, helps you manage the overall project, which might include multiple document files and other components. Its primary purpose is to allow you to manage your MVP file easily. Project Editor is your entry point for creating the project and for each editing session.

Project Editor's menu items let you

- ü Define the windows and panes you will use in your application
- ü Define topic groups and the associated entry and exit commands
- ü Define the copyright citations to be displayed
- ü Select the icon to be displayed for your application
- ü Define commands to be executed when your file is loaded
- ü Define options controlling Search functions, such as data types and keyword indexes
- ü Set compiler options such as paths, compression, and CD-ROM optimization
- ü Compile your application
- ü Review any error messages generated by the compiler
- ü Run Viewer to display the compiled file

A short tour of some common Project Editor dialog boxes follows, showing features that are used throughout the Viewer authoring system. Figure 2–1 shows the main window of Project Editor. This is used to maintain the list of document and Baggage files used by your application, and to choose functions from the menus.

```
{ewc vwrht2, TsTextButton, "Figure  
2-1" [Macro=JI('viewerht.mvb>SecWin', 'fig2_1')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Figure 2–2 shows the choices available within Project Editor's Section menu. These are used to modify sections of the project file. You will use some of these choices often. Figures 2–3 through 2–6 show dialog boxes that are displayed after choosing items from the Section menu.

```
{ewc vwrht2, TsTextButton, "Figure  
2-2" [Macro=JI('viewerht.mvb>SecWin', 'fig2_2')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Figure 2–3 shows the Window Properties dialog box, which is used to define characteristics of main or secondary windows. This is a typical dialog box in the Viewer authoring system. It includes a check box, several fields

with pull-down lists of values that can be selected, and several fields that must be entered by typing the desired values.

```
{ewc vwrht2, TsTextButton, "Figure  
2i_1/23"[Macro=JI('viewerht.mvb>SecWin', `fig2_3')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Figure 2-4 shows the [CONFIG]—Configuration Script dialog box, which is used to define commands to be executed when the application is loaded. The Paste Command button provides a list of Viewer commands that can be added to the script. The Edit Command button displays a dialog box tailored to the selected command, with separate fields for entering each parameter. Pull-down lists are provided for many fields to prevent misspelling or other errors.

```
{ewc vwrht2, TsTextButton, "Figure  
2i_1/24"[Macro=JI('viewerht.mvb>SecWin', `fig2_4')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Figure 2-5 shows the [FTINDEX]—Data Types dialog box, which is used to define data types and associated characteristics that control the Search operation. Three fields, which require file names to be entered, have a box with an ellipsis (...) after the field. Clicking on this box displays a file selection dialog box that is very similar to the Windows standard File Open dialog box. This dialog box allows the file name to be selected from the actual contents of the disk to prevent misspelling or other errors.

```
{ewc vwrht2, TsTextButton, "Figure  
2i_1/25"[Macro=JI('viewerht.mvb>SecWin', `fig2_5')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Figure 2-6 shows the [GROUPS]—Groups dialog box, which is used to define groups of topics. The New button is used to add a new group. Entry and Exit scripts, containing commands to be executed when the group is entered or exited, can be defined much like the CONFIG script shown in Figure 2-4. Many common dialog boxes, such as those used to define scripts, paste commands, or edit commands, are used throughout the Viewer authoring system. This makes it easier for you to use the system by reducing the number of components to be learned.

```
{ewc vwrht2, TsTextButton, "Figure  
2i_1/26"[Macro=JI('viewerht.mvb>SecWin', `fig2_6')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Project Editor makes it easy to edit your document files or any of the files listed in the Baggage section. If you double-click on one of these names, Project Editor will start the program associated with that file's extension and open the file to be edited. This is usually the editor for that type of file. For example, you can start Word and open a document by double-clicking on the

name of the RTF file. If the sound or movie file extensions are associated with programs to play the files (such as MPlayer) instead of editor programs, you should change those associations. That can be done easily through the Windows File Manager.

### The Viewer Topic Editor

Topics are the primary building blocks of your document files. A topic represents a section of text, pictures, hot spots, and multimedia displays.

Topic Editor is used to help you define topics, build commands, and use special-purpose formatting in your document files. It provides menus and special dialog boxes to make this process as simple as possible. These dialog boxes use information from your project file to allow many choices to be made by selecting an entry from a list. This eliminates many opportunities for typographical errors that occur when users must type in their choices.

Project Editor starts both Topic Editor and Word when you double-click on the name of a document file. You invoke Topic Editor through a hot key which you define in Project Editor. Topic Editor starts by presenting you with a list of commands or operations available. Figure 2-7 shows the first part of this list. The How-Tos in this book demonstrate many of these commands and operations, and give you the background needed to understand most of the others. All of the commands and operations are described in the Viewer documentation.

```
{ewc vwrht2, TsTextButton, "Figure  
2i½7"[Macro=JI('viewerht.mvb>SecWin', `fig2_7')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Once you select the desired function, you are presented with one or more dialog boxes where you select the appropriate options or enter necessary values. A short tour of some common Topic Editor dialog boxes follows, showing features that are used throughout the Viewer authoring system. The meaning of the options shown in these dialog boxes are demonstrated in the How-Tos in this book.

Figure 2-8 shows the dialog box used to define the characteristics of a text hot spot. This is typical of the dialog boxes displayed after a command is chosen in the New Viewer Element dialog box. The elements of the command or operation are listed on the left, with one element selected. The right side is tailored to the selected element, showing the appropriate fields and options. This dialog box uses radio buttons, text boxes, and a pull-down list. Pull-down lists in Topic Editor often reflect components that were defined in Project Editor, such as the names of windows or topic groups.

```
{ewc vwrht2, TsTextButton, "Figure  
2i½8"[Macro=JI('viewerht.mvb>SecWin', `fig2_8')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Figure 2-9 shows the dialog box that is displayed when you choose Hidden Text is Command(s) in the dialog box shown in Figure 2-8. The fields on the right side are changed to show a command script with Edit Command and Paste Command buttons. These buttons display the same

dialog boxes as their Project Editor counterparts seen in Figure 2–4.

```
{ewc vwrht2, TsTextButton, "Figure  
2½9"[Macro=JI('viewerht.mvb>SecWin', `fig2_9')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Figure 2–10 shows the dialog box that is displayed when you click on an Edit Command button. It shows the name of the command being edited and the fields required by that command. Pull-down lists are provided where appropriate.

```
{ewc vwrht2, TsTextButton, "Figure  
2½10"[Macro=JI('viewerht.mvb>SecWin', `fig2_10')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Figure 2–11 shows the Multimedia Options dialog box that is displayed when you choose the Multimedia item from the New Viewer Element dialog box shown in Figure 2–7. The options available through this dialog box allow you to specify playing any file supported by the Windows Multimedia Command Interface (MCI), including Wave and MIDI sound files, AVI movies, and animation files. This dialog box lets you specify standard or customized buttons to be displayed that let the user start, stop, rewind, and otherwise control the file playback. This complex and vital subject, which requires a full chapter in the *Viewer Authoring Guide*, can be controlled through this dialog box.

```
{ewc vwrht2, TsTextButton, "Figure  
2½11"[Macro=JI('viewerht.mvb>SecWin', `fig2_11')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Topic Editor inserts the appropriate command, or sets the appropriate formatting options, in your Word document when you click on OK to accept the values you specified. In some cases it also updates the project file. For example, Topic Editor lets you specify that a multimedia file you selected to play should be stored in Baggage. If you check that option, Topic Editor adds the name of that file to the Baggage section of your MVP file.

If you select the text of a previously created Viewer command and invoke Topic Editor, it bypasses the initial window and displays the appropriate dialog box to edit that command. That makes it easy for you to revise commands later on. If you select an entire topic, Topic Editor displays a list of all the commands and lets you edit any or all of them.

## 2.4 What are the Support Tools?

Brief summaries of the functions of Viewer's useful utilities follow. All of these programs are described in detail in Viewer's documentation.

### **Bitmap Editor (vflid137438953482)BitEdit(vflid18013857343602688)**

BitEdit, as you can see from its name, is designed to help you edit bitmap picture files. It contains standard painting tools, as does Windows Paintbrush, but is designed to supplement, rather than replace, standard painting programs. Its biggest strength is its ability to work in tandem with the Palette Editor to adjust the palette used by a picture. Figure 2-12 shows the BitEdit window and floating control bar, with a picture file loaded and ready to be edited.

```
{ewc vwrht2, TsTextButton, "Figure  
2i½12"[Macro=JI('viewerht.mvb>SecWin', `fig2_12')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

BitEdit is capable of reading a large number of file formats, including many that cannot be used by Viewer. This allows it to convert files to acceptable formats while adjusting the colors, palette or other characteristics. BitEdit can also write files that Viewer cannot accept, such as Apple PICT and Microsoft RLE DIB files.

BitEdit fully supports 16- and 256-color (4- and 8-bit) pictures. True-color images (with 65,536 or 16.7 million colors) are converted to 256 colors when they are loaded.

### **Palette Editor (vflid137438953482)PalEdit(vflid-9078975914968088576)**

If your Windows system is running in 256-color mode, that means that it can display 256 of the 65,536 possible colors at one time. The programs in use select which of those possible colors will be used. There is one active palette at all times, containing the set of up to 256 colors that can be used at that moment. That palette applies to all visible windows.

Bitmap pictures and video images contain their own palettes, representing the colors used in those files. When the images in these files are displayed, the palette becomes the active palette for the entire display. All other windows are redrawn using the nearest colors that are present in the new palette. This can cause dramatic and unexpected results if there are no similar colors! Undesirable results can also occur if you display two images at once that use different palettes.

The Palette Editor allows you to adjust palettes, and copy a palette from one image to another, to prevent or minimize such problems. It allows you to control the color changes to minimize the visual impact. This can be far better than letting the software on your users' systems select the colors!

The Palette Editor can work in conjunction with the Viewer Bitmap Editor as shown in Figure 2-13. This is often the most useful and powerful way to use the Palette Editor.

```
{ewc vwrht2, TsTextButton, "Figure  
2i½13"[Macro=JI('viewerht.mvb>SecWin', `fig2_13')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

### **Wave File Editor**

#### **{\vflid137438953482}WaveEdit{\vflid18013857343602688}**

The Wave File Editor allows you to create a new Wave sound file, edit an existing file, or copy sounds between files. Portions of a recording can be cut, copied, or combined (mixed). Periods of silence can be inserted, selections can be faded up or down, and the volume can be changed.

This utility also allows the sample size or frequency of a file to be reduced, shrinking the file at the expense of a reduction in the quality of the sound. The difference may not be apparent to the listener if the original recording used the highest sampling level.

The Wave contents are displayed as a graphic showing the sound volume over time, as shown in Figure 2–14.

```
{ewc vwrht2, TsTextButton, "Figure  
2i½14"[Macro=JI('viewerht.mvb>SecWin', `fig2_14')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

#### **File Conversion ({\vflid137438953482}Convert{\vflid-9078975914968088576})**

This utility allows you to convert unsupported sound, picture, and palette files to formats that can be used by Viewer. Both Wave and MIDI sound file formats can be converted. Files can also be converted to some unsupported formats, such as Microsoft RLE DIB, if this is needed.

The main window for the Viewer Convert utility is shown in Figure 2–15.

```
{ewc vwrht2, TsTextButton, "Figure  
2i½15"[Macro=JI('viewerht.mvb>SecWin', `fig2_15')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

### **Segmented Hot spot Editor**

#### **{\vflid137438953482}Shed2{\vflid18013857343602688}**

Shed2 (shown in Figure 2–16) is designed to allow you to define multiple hot spot regions within a single picture. It provides a simple method for selecting and defining the action to be taken for each individual hot spot region. The updated file is saved with a SHG extension. These files can be displayed by the Viewer commands, but they are not supported by other programs.

The picture used in an existing SHG file can be replaced by a new version without redefining all of the hot spots. This was not supported in earlier versions of Shed, used with Viewer 1.0 and the Windows Help system.

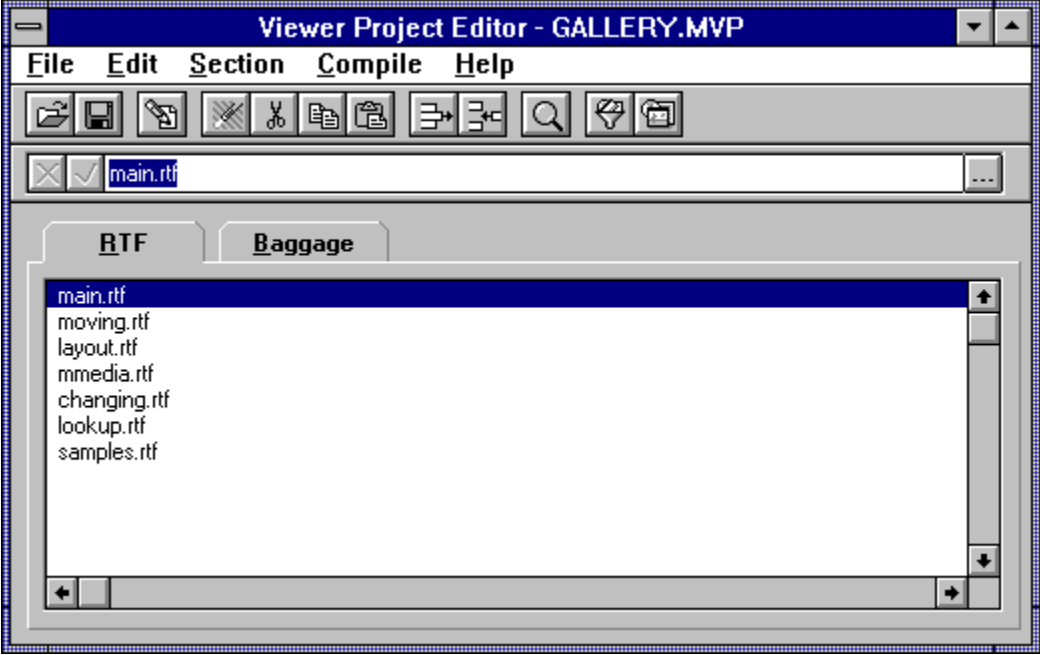
The picture should be displayed with a caption or associated text that will help inform the user of the picture's purpose. The only visual clue the user of your application sees to recognize the existence and location of these hot spots is the appearance of the cursor. The cursor changes to a pointing finger while over one of the hot spot regions, just as it does for other hot spots.

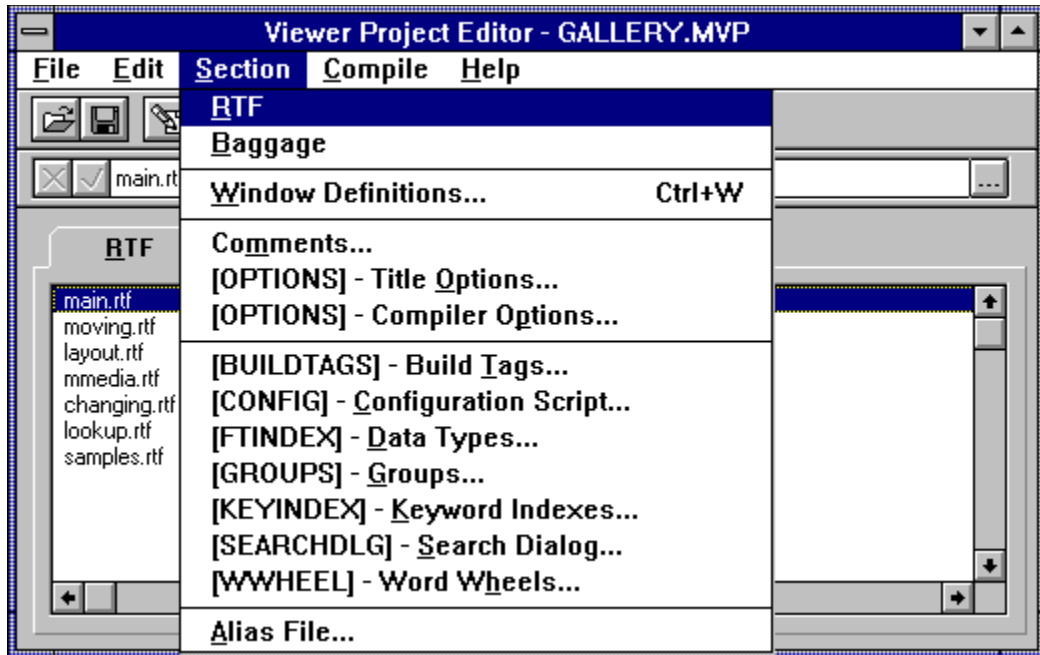
```
{ewc vwrht2, TsTextButton, "Figure  
2i½16"[Macro=JI('viewerht.mvb>SecWin', `fig2_16')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

## 2.5 Tips and Tricks

- ⇒ The Viewer compiler requires that you have a local hard drive on your computer. A diskless system on a {vfld137438953484}LAN{vfld4926370438184960} cannot be used.
- ⇒ The compiler work files are placed in the drive and directory pointed to by your SET TEMP parameter (See Other Files Used to Create an Application, in section 2.2). This should never point to a remote drive on a LAN—this can cause extremely long compile times.







**Window Properties**

**Window Name:**

**Top:**       **Height:**

**Left:**       **Width:**

**Background Color:**  ...  **Use Default Color**

**Background Picture:**  ...

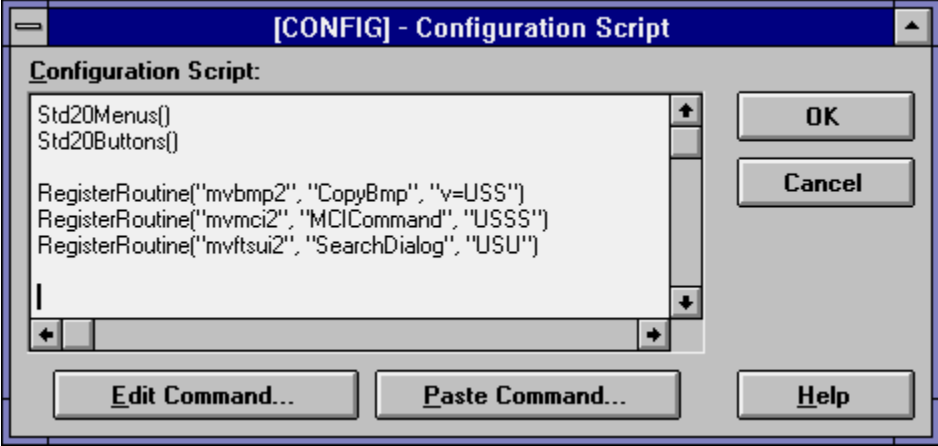
**Window Caption:**

**Initial State:**  ▾

**Stay On Top:**  ▾

**Minimize with MAIN:**  ▾

**Coordinate System:**  ...



[FTINDEX] - Data Types

**Search Data Types:**

- 0 - Words
- 1 - Number
- 2 - Date
- 3 - Time
- 4 - Epoch

**Data Type Number:**  
0

**DLL Filename:**  
mvbrkr2.dll

**DLL Routine Name:**  
FBreakWords

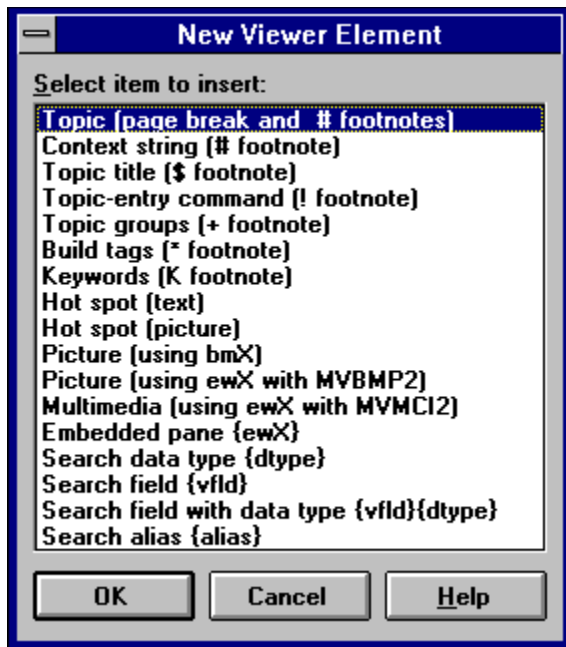
**Stop-Word List Filename:**

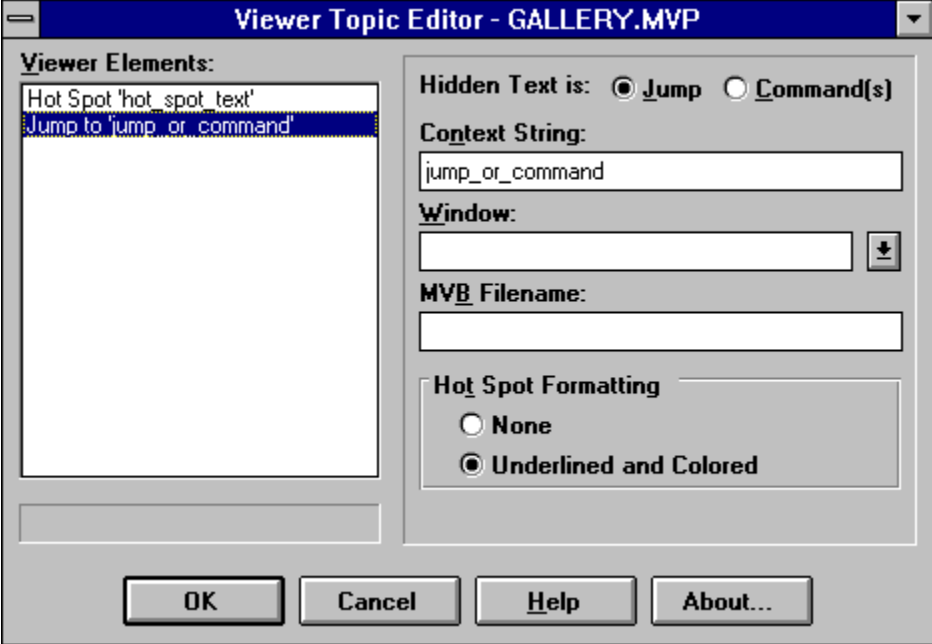
**Character Tables Filename:**

**Comment:**  
reakAndStemWords to use word stemming

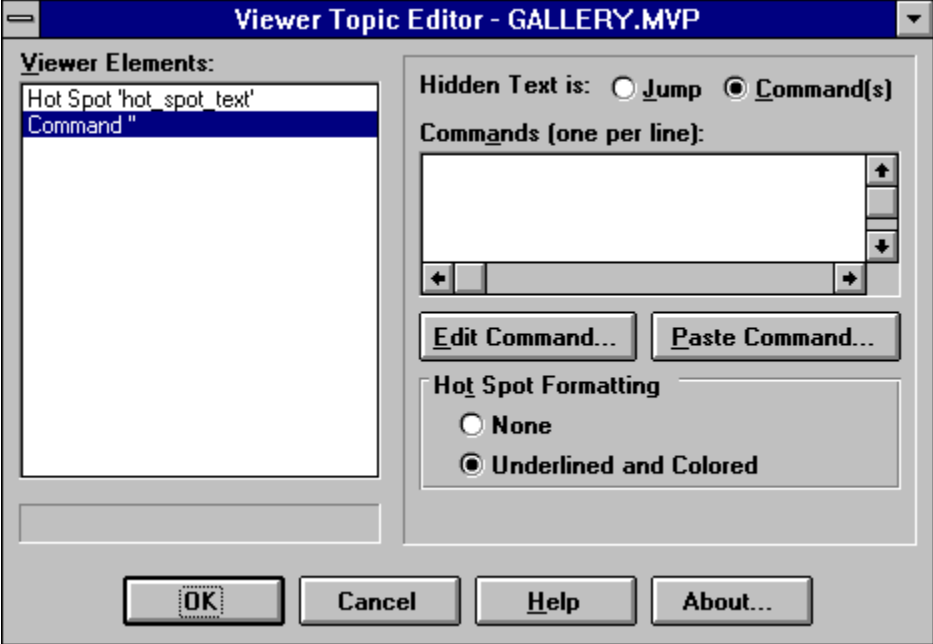
**Delete**   **New**   **OK**   **Cancel**   **Help**











**Edit Command**

**Edit Command:** CreateButton

**ButtonID:**

**ButtonCaption:**

**Command:**

ButtonID: Specifies the identifier for the button

**Multimedia Options**

**Media**

MCI Device: WaveAudio [v] [↓] Preview

Filename: [ ] [⋮] Edit File

Store File in Baggage

**Playback Options**

Looping

Auto-Start

Share As: [ ]

**Position**

Left

Right

Text-Aligned

Layout...

**Range**

|        | Track # | Milliseconds |
|--------|---------|--------------|
| Start: | [ ]     | [ ]          |
| End:   | [ ]     | [ ]          |

Play Entire File

Edit Sections...

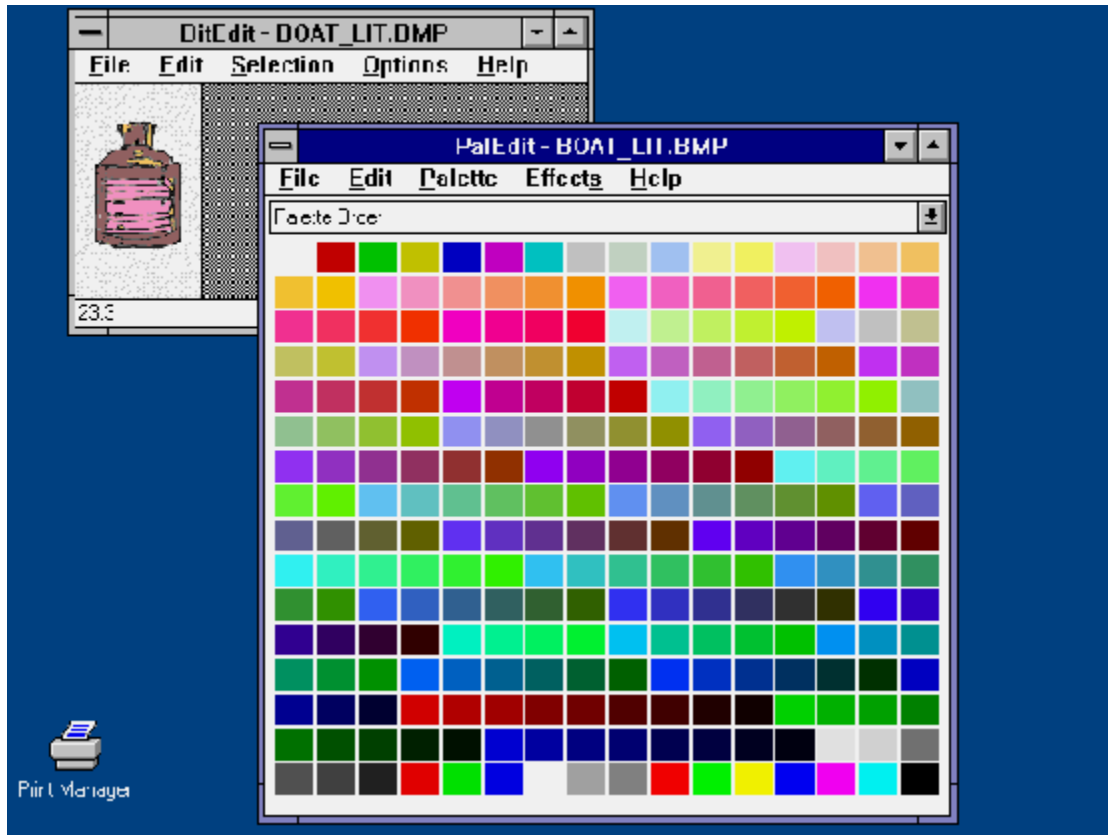
**Controller**

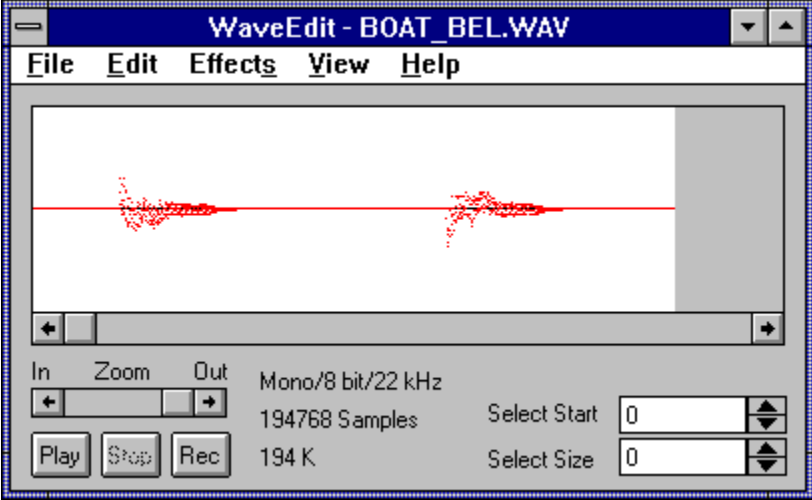
Show Controller

Edit Controller...

OK Cancel Help







**Convert**

Help

**Source**

Type: **Microsoft Windows DIB** ▾

File(s):  
C:\MVPUBKIT\MVSAMPLE\GALLERY\\*  
C:\MVPUBKIT\MVSAMPLE\GALLERY\\*

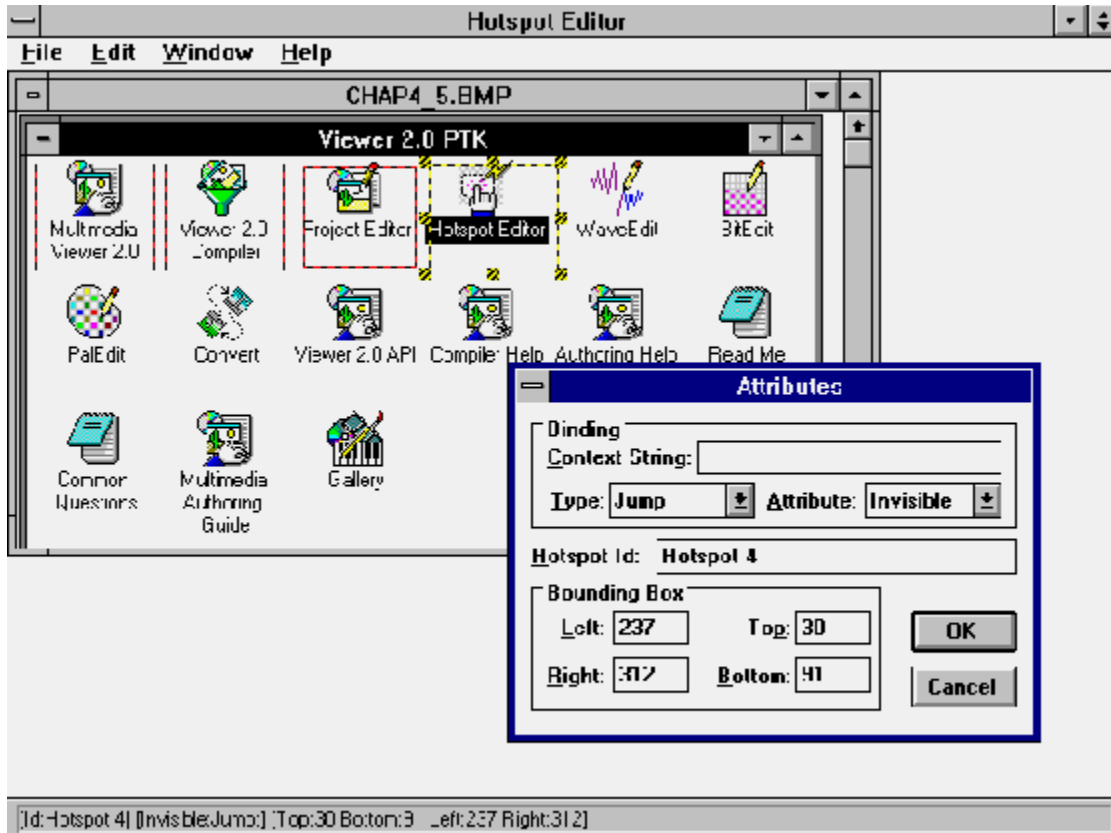
Add...  
Remove

**Destination**

Type: **Microsoft RIFF RLE DIB** ▾      Extension: **rdi**

Dir: C:\MVPUBKIT\MVTOOLS

Set Dest Dir...      Convert







This chapter teaches you the common operations necessary to create a Viewer application using the major components of the Viewer authoring system—Project Editor and Topic Editor. These programs work together to make it easy for you to use most of the features of Viewer even if you aren't a technical wizard. They let you concentrate on being an *author*—creating the text, sound, and images that make up your application—without having to get involved in the details of how it works.

If you are already using Viewer, you should skim through this chapter even if most of the material is already familiar to you. Some of the tips and suggestions may be new to you.

If you are new to Viewer, but have created Windows Help files, you should read this more carefully. Even though many of the technical details are the same, the differences are vital. You'll be impressed by how easy the Viewer authoring system is to use—even if you have used one of the powerful Help authoring packages.

If this is your introduction to Viewer, and you haven't created Help files, you need to study this material carefully. This chapter provides the foundation that the following chapters build on.

This chapter describes the initial configuration of Viewer, and some customizing you may want to do in Word for Windows. It also explains some procedures and documentation techniques that will help you avoid problems later on. It covers virtually everything required to use Viewer as an application Help system instead of WinHelp.

This chapter is designed for you to work through each of the How-Tos in order. They all use the same files in the same directory, and each one adds to the results of the previous section. The files that result from each section are included on the enclosed CD-ROM in individual directories to let you compare your results. These files could also be used to provide the proper starting point if you decide to skip some sections.

## Start a New Project?

Complexity: EASY

### Problem

I'm ready to start developing a new application in Viewer. How do I start?

### Technique

First, you use the Windows File Manager to set up a useful directory structure for the files you create in this How-To. This standard structure is repeated in all of the other How-Tos in this book. It is also used on the enclosed CD-ROM, which makes it easier for you to compare your results to mine.

If this is your first application, you need to set some configuration preferences in Project Editor. Then you use Project Editor to create your project file (MVP file). You won't be setting any options or definitions yet, other than the name of your text file.

The files created in this How-To may be found on the enclosed CD-ROM disk in the VIEWERHT\HOWTOS\CHAP3\CHAP3\_1 subdirectory.

### Steps

1. As explained in Chapter 2, your first step *must* be designing your application. If you don't start off with a plan for your user interface, window layout, actions, topic grouping, searching, and all the other parts of your application, you end up redoing your work many times. Not only do you waste a lot of your own time, but the end result isn't nearly as good either. Assume that all of the planning has taken place. Later material helps you understand how to do the planning for your own projects.

Create the directories where you store the files for this How-To:

2. Start File Manager, and select your root directory.
3. Choose Create Directory from the File menu.
4. Enter VIEWERHT as the name of the high-level directory where you hold all of these How-Tos. Click on OK.
5. Select the VIEWERHT directory.
6. Choose Create Directory from the File menu again.
7. Enter CHAP3 as the name of the directory for this How-To. In all later chapters, you use the chapter and section numbers to name the directories where you create the files for the How-Tos. Each directory at this level contains the project file and compiled Viewer file for the corresponding How-To.
8. Click on OK.
9. Select the How-To directory, CHAP3.
10. Choose Create Directory from the File menu once again.
11. Enter TEXT as the name of the directory where you store your document files. Click on OK.

12. Repeat steps 9 through 11, creating the following subdirectories for your How-To: **SOUNDS**, **PICTURES**, and **MOVIES**. These are where you store your sound files, picture files, and movie or animation files. Figure 3-1 shows the tree structure you should have.

```
{ewc vwrht2, TsTextButton, "Figure  
3i½1"[Macro=JI('viewerht.mvb>SecWin', `fig3_1')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

13. Close the File Manager, and start Project Editor.

Now make sure Project Editor is configured properly:

14. If you haven't done this yet, choose Preferences from the Edit menu to display the Preferences dialog box.
15. Make sure the paths shown for Word for Windows and the Viewer compiler are correct. Change them if necessary.
16. Set the {vfld137438953482} hotkey {vfld-9223348397354778624} you want to activate Topic Editor. I suggest following the Microsoft recommendation of **[CTRL]-[T]**. The completed dialog box should look like Figure 3-2.

```
{ewc vwrht2, TsTextButton, "Figure  
3i½2"[Macro=JI('viewerht.mvb>SecWin', `fig3_2')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

17. Click on OK to save these values. You don't have to do this again unless you want to change one of the choices.

Now create the initial project file:

18. Choose Save As from the File menu.
19. Select the subdirectory you created for this How-To (\VIEWERHT\CHAP3).
20. Enter the How-To name, **CHAP3**, as your file name. The MVP extension is added if you leave it off. Click on OK.

Next create the text file:

21. Be sure the RTF file folder tab is on top. If not, click on it to bring it to the top.
22. Choose Insert Line from the Edit menu. The insertion point appears in the edit line.
23. Type **TEXT\CHAP3.RTF** and press **[Enter]** or click on the box containing the "X". The completed window should look like Figure 3-3.

```
{ewc vwrht2, TsTextButton, "Figure  
3i½3"[Macro=JI('viewerht.mvb>SecWin', `fig3_3')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

24. Choose Save from the File menu to save the information in your project file.

### **How It Works**

Each project should have its own directory, with standard subdirectories for the different types of files. It's a good idea to always create each of these subdirectories, even if you don't need them all. This consistency costs you almost nothing in time, effort, or disk space, but it helps you avoid problems later if you decide to include a file that you hadn't planned for.

You have to specify the text subdirectory with the name of your RTF file, such as TEXT\CHAP3.RTF, for Project Editor to create the file in the right place. Don't specify the name of the project directory—all paths are considered to be relative to the project directory. The project directory is always the directory where the project MVP file is located. You select this directory when you do your first Save As in Project Editor.

Project Editor is the master control point of the Viewer authoring process. As you will find in the following How-Tos, you always start from there. You should explore the menu items and see if you can anticipate what they do. Be sure your project file is saved first, and don't save any changes you make while experimenting.

### **Comment**

You create the text file in the next How-To, before creating the first topic.

Project Editor has standard default values for most of the options you could select. You see the effect of those if you examine your CHAP3.MVP file in Notepad. This file is formatted just like a standard Windows INI file. It is divided into sections, with the names of each section enclosed in square brackets. Within each section are individual parameters followed by an equal sign and the assigned value. You rarely need to work with this file directly, but you should become familiar with its appearance. It helps you understand how the system is working, which can help you to make the best use of Viewer.

## Create a Contents Topic?

Complexity: EASY

### Problem

I need to begin creating my document file by setting up a table of contents for my application.

### Technique

You use Project Editor, Topic Editor, and Word for Windows any time you are working with your text file. Project Editor and Topic Editor let you define the technical information easily, through dialog boxes.

The table of contents in a Viewer application serves the same purpose as the table of contents in a book. It provides a list of the subjects and a way to find each of them. Books print the page number for each subject so the reader can go to the desired page. Viewer applications define the subject names as hot spots, so the user can go to a subject by clicking on its entry in the table. You create such hot spot entries for each subject.

Before you can create the table of contents, you need to know how the application will be organized. (How else would you know what to put in the table?) You also need to know what names, known as `{vfld2305858952132296714}context strings{vfld-9079242005371944960}`, to assign for Viewer to use to refer to each topic. You use those names to tell Viewer which topic to display when the user clicks on a hotspot.

You usually need to know the context strings for topics before you've created the topics themselves. That's true in this case—you use context strings in the table of contents. The easiest way to do this is to develop your own naming convention, and to keep a list of the topics you plan to create with their context strings and where these context strings are referred to. A common naming standard is simple: *ctx\_* (for "context") followed by an abbreviated topic title. You can use any system you want. Viewer doesn't care about upper- or lowercase letters in a context string—it considers *Abc*, *ABC* and *abc* as the same.

The files created in this How-To may be found on the enclosed CD-ROM disk in the VIEWERHT\HOWTOS\CHAP3\CHAP3\_2 subdirectory. This How-To builds on the file created in How-To 3.1, located on your hard disk in subdirectory VIEWERHT\CHAP3.

### Steps

Design and plan your table of contents:

1. You need to know the name of each subject you want to include and how you want the subjects to appear. Should some be indented, showing subject groupings? Should they be numbered? For this How-To, use the descriptions as shown in Table 3-1.
2. You also need to identify which topic to display for each subject listed in the contents, and the context string for each of those topics. For this How-To use part of the outline for chapters 2 and 3 of this book.

**Table 3–1 Planned Topics and Context Strings**

| Description                          | Context String  |
|--------------------------------------|-----------------|
| Designing a Viewer Application       | ctx_chap_2      |
| Introduction                         | ctx_2_intro     |
| How Should I Design the Application? | ctx_2_design    |
| What Files Can Be Used?              | ctx_2_files     |
| What Are the Authoring Tools?        | ctx_2_authoring |
| What Are the Support Tools?          | ctx_2_support   |
| Creating a Simple Application        | ctx_chap_3      |
| Introduction                         | ctx_3_intro     |
| Start a New Project                  | ctx_3_project   |
| Create a Contents Topic              | ctx_3_contents  |
| Create Additional Topics             | ctx_3_addl      |
| Create a Popup Topic                 | ctx_3_popup     |
| Create Searchable Keywords           | ctx_3_keywords  |
| Keep the Table of Contents Visible   | ctx_3_visible   |
| Tips and Tricks                      | ctx_3_tips      |

3. Usually you would prepare your directories and files next, but in this chapter continue to use the the directories and files from the previous section.
4. Use the File Manager to copy the CHAP3.ICO file from the VIEWERHT\HOWTOS\CHAP3\CHAP3\_2\PICTURES directory on the enclosed CD-ROM to the VIEWERHT\CHAP3\PICTURES subdirectory on your hard drive.
5. Start Project Editor, and open your project file, \VIEWERHT\CHAP3\CHAP3.MVP.

Now you're ready to start creating the text file and topic:

6. Double-click on the text file name. This is how you indicate that you want to edit the contents of a file. Because this file does not exist, Project Editor displays a message box saying "RTF file C:\VIEWERHT\CHAP3\CHAP3.RTF does not exist. Create it?"
7. Click on the Yes button.
8. Project Editor starts Word for Windows. If Word doesn't start, your entry for the Word path in the Edit Preferences dialog box is probably incorrect. Fix it and try these steps again.
9. Word displays the Convert File dialog box, showing that the selected file is in Rich Text Format (RTF). Click on the OK button.
10. Your document appears. So far, it is completely empty. Let's do something about that!
11. Before you start entering any text, click on the paragraph symbol, also known as a carriage return symbol , at the right-hand end of the Word ruler. This causes all {vfld137438953482}hidden text{vfld-3532125269655520}, blanks, and control codes to be displayed.

It is usually much easier to create your document with these codes visible. You can click on that symbol again any time you want to see how your document looks without codes.

The only time it's easier to work with this option off is when you use tabs to line up text that includes hot spots. Because hot spots include hidden text, they can throw off the apparent effect of tabs.

12. Use the hotkey you selected earlier to bring up Topic Editor. This displays the New Viewer Element window shown in Figure 3–4, which lists the commands you may use.

```
{ewc vwrht2, TsTextButton, "Figure  
3i½4"[Macro=JI('viewerht.mvb>SecWin', `fig3_4')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

13. Select Topic (Page break and # footnotes) and click on OK. This displays the Viewer Topic Editor window. The left side lists the elements of the selected command, and the right side shows information about the selected element.
14. Enter the desired context string for this topic as `ctx_contents`. The completed dialog box should look like Figure 3–5.

```
{ewc vwrht2, TsTextButton, "Figure  
3i½5"[Macro=JI('viewerht.mvb>SecWin', `fig3_5')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

15. Click on OK.
16. Notice that a hard page break was inserted above your footnote code, even though this is the beginning of the document. That break is not needed, and can cause confusion later. Delete it now.
17. Now use your hotkey to bring up Topic Editor again, and select the Topic Title (\$ Footnote) entry.
18. Enter **Table of Contents** as the title, then click on OK.
19. Your document should now show two footnote codes: # and \$ .
20. Type in your topic heading as **Contents**. Format this word with a larger than normal font, and use boldface to make it stand out.
21. Next create the first hot spot. Drop down a few lines and type in the text for the first entry:

#### Designing a Viewer Application

22. Select the text you just typed in, and bring up Topic Editor with your hotkey. See Figure 3–6.
23. Notice that Figure 3–6 shows a much different list of commands. This list is shown when you select text before you bring up the Editor. In this case you want the first entry, Hot spot (text), which is already selected.



```
{ewc vwrht2, TsTextButton, "Figure  
3i½6"[Macro=JI('viewerht.mvb>SecWin', `fig3_6')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

24. Click on OK to display the Viewer Topic Editor window shown in Figure 3–7.

```
{ewc vwrht2, TsTextButton, "Figure  
3i½7"[Macro=JI('viewerht.mvb>SecWin', `fig3_7')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

25. When the hot spot text element is selected, this window shows a choice of Hot Spot Type. Leave it as Normal.
26. Select the Jump to element to display the hot spot options.
27. Enter the context string for this entry as `ctx_chap_2`.
28. Leave the Window and MVB Filename entries blank.
29. Leave the Formatting option at the default value of Underlined and Colored. This produces the standard appearance. The result should look like Figure 3–8.

```
{ewc vwrht2, TsTextButton, "Figure  
3i½8"[Macro=JI('viewerht.mvb>SecWin', `fig3_8')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

30. Click on OK to return to Word.
31. Without moving the insertion point, choose Character from the Format menu and look at the current character formatting options. You see that Hidden is selected.
32. Turn off Hidden and click on OK. If you do not do this, the carriage return and text you are about to enter will be invisible!
33. Now enter the rest of the hot spots chosen in step 2. Type in the text, and create the hot spots.
34. When you are done, your document should look like Figure 3–9.

```
{ewc vwrht2, TsTextButton, "Figure  
3i½9"[Macro=JI('viewerht.mvb>SecWin', `fig3_9')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

35. Now that you have completed your changes to the document, save your file and exit Word. Choose Save As from the File menu.
36. Select Rich Text Format (\*.RTF) under Save File as Type.
37. Click on OK.
38. Choose Exit from the File Menu or double-click on the Windows control menu box to exit Word.
39. When the message box “Do you want to save changes to CHAP3.RTF”

appears, you *must* click on No. If you click on Yes, Word saves your document in its DOC format using your {vfld137438953484}RTF file{vfld8749649109884862464} name. This format cannot be used by the Viewer compiler; if you click on Yes by accident, error messages will appear when you compile. If you click on Yes, additional dialog boxes are displayed that let you correct this mistake.

Next define the context string of your Contents topic:

40. You should be back in Project Editor. If not, switch to it by holding **[CTRL]** and pressing **[TAB]** until Project Editor appears.
41. Choose Title Options from the Section menu and enter the context string of your Contents topic as `ctx_contents`.
42. In the same dialog box, you can attach an icon to this application. Click on the box with the ellipsis (...) after the Icon Filename field to display the Icon File dialog box, which looks and works just like the standard File Open dialog box. Select the PICTURES\CHAP3.ICO file, then click on OK. The completed dialog box should look like Figure 3–10. Click on OK to return to Project Editor’s main window.

```
{ewc vwrht2, TsTextButton, "Figure  
3½10"[Macro=JI('viewerht.mvb>SecWin', `fig3_10')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

43. Now you can compile the application. Choose Build from the Compile menu.
44. Because you just changed the project file, before compiling Viewer asks “Save changes to CHAP3.MVP before starting compiler?” Click on Yes.
45. The compiler window shows a progress meter bar, error messages, and a description of the current activity while it works, as shown in Figure 3–11. The handle next to the project name turns while the compiler is running.

```
{ewc vwrht2, TsTextButton, "Figure  
3½11"[Macro=JI('viewerht.mvb>SecWin', `fig3_11')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

46. The error messages report that the context strings in your hot spots have not been defined. In this case you already know that, so it’s OK. You can examine the error messages by choosing Display Error Log from the Compile menu.
47. Now test your application by choosing Run Viewer on CHAP3.MVB from the File menu. Your window should look like Figure 3–12.

```
{ewc vwrht2, TsTextButton, "Figure  
3½12"[Macro=JI('viewerht.mvb>SecWin', `fig3_12')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

48. Minimize the window to an icon—you see the new icon associated with this file.

### **How It Works**

Project Editor creates an RTF file if one doesn't exist when you try to edit it. The advantage to this is that Project Editor always starts Word with the name of your file. If the file already exists, Word just loads it and proceeds. When you save the file, Word already has the right file name. This doesn't work nearly as cleanly if the file doesn't exist when Word is started.

Topic Editor displays an appropriate list of commands in the New Viewer Element window based on whether or not any text is selected. If you select an existing command or hot spot, Topic Editor bypasses the New Viewer Element display and goes directly to the Topic Editor window, where you can edit your previous choices.

The Topic Editor window always lists the elements for the selected command on the left, and options for the selected element on the right. An Options button is provided if a complete separate dialog box is used to configure an element. The selected text, such as when defining a hot spot, cannot be changed in this window. It can only be changed in Word.

A standard text hot spot has a very simple structure, as you saw in this How-To. The text that is displayed in the runtime application is formatted with a double underline. It is followed, without any spaces between, by the context string of the topic that should be displayed if the user clicks on this hot spot. The context string is formatted as hidden text. If you wanted to, you could create these easily without using Topic Editor by entering the desired text with the proper character format directly into your document. Topic Editor makes it easier to create these and reduces the chance of errors, so you should use Topic Editor unless you have a very good reason to enter context strings directly.

Your document must be saved in Rich Text Format (RTF) to be usable by the compiler. Word always saves in DOC format if you use the File Save menu item. This is why you must use the Save As item, and specify the RTF format. When you exit Word, it asks if you want to save your file because you didn't use the standard Save command. If you click the Yes button, Word will resave your file in DOC format, but using your RTF file name.

If you try to test any of your hot spots, you get an error message because the referenced context strings don't exist yet. You create those topics and context strings in the next How-To.

### **Comment**

Planning and documenting the technical details of your project, such as the context strings, is more important than may be apparent to you at this stage. A few important points to understand:

- ü You are often creating topics, and creating hot spot references to those topics, at much different times. The only way to assure that you always use the right context string is to build a list of your topics and context strings.
- ü The compiler reports context strings used in hot spots that are not

defined. It does not report undefined context strings used in commands, or topics that are never referenced. Most important, it cannot detect when you use the context string for the wrong topic in a hot spot. Good documentation can help prevent such problems.

- ü Context strings must be unique within a project. Remember that Viewer ignores the case of your string.

When your file is loaded, Viewer displays the first topic in the file unless you specify a contents topic through the Title Options choice of Project Editor's Section menu. Setting the contents topic also enables the Contents button on the Viewer user interface to display the right topic.

You delete the unnecessary hard page break at the beginning of the document because Viewer reports most errors with the relative {vfld137438953484}topic number {vfld-9006918320930160640}. That extra page break would result in an extra topic to the compiler. This is likely to throw you off when looking at your error log: "What do you mean, error in Topic 2? I only created one topic!" You can avoid the extra page break by selecting Context String (\$ footnote) for the first topic.

One common cause of problems is letting the hidden character formatting used in a hot spot remain in effect through following carriage returns or text (see steps 34 and 35). Hidden

{vfld137438953484}carriage returns {vfld72057052872048640} are reported by the compiler, and then ignored. The following text appears on the same line. Hidden carriage returns are not apparent while in Word—you have to mark the carriage return symbol and bring up the Format Character dialog box to see them. Hidden text is apparent from the dotted underline while in Word.

Another common cause of problems is misspelling {vfld137438953484}context strings {vfld2850492484943872}, or including one or more leading blanks with the name. Personally, I find that I often leave the initial "j" from the default contents of this field in Topic Editor. I miss it when highlighting the text to replace.

You may find it easier to use {vfld137438953482}keyboard shortcuts {vfld-9223348397354778624} than mouse commands while you are typing. You can select a line of text with **[SHIFT]–[END]**, bring up Topic Editor with your hotkey, then use the **[→]**, **[TAB]** and **[ENTER]** keys to move to the proper option within a dialog box. This can also prevent many of the common errors.

When you have a series of hot spots to enter, it is often easier to first enter all the displayed text, and then use Topic Editor to create the hot spots. This avoids the need to turn off the formatting after each line. The alternative is to use **[CTRL]–[SPACE]** to turn off the formatting after each line before proceeding.

|  |
|--|
| <p>If you are creating a very large application, you may prefer to save your file in DOC format between compiles. This is much faster than creating an RTF file, and also produces a much smaller file. Save it using the RTF file name. This lets you continue to use the Project Editor interface, and you won't risk having an obsolete version of the file in the RTF name (as you might if you used different names). If you do this, be sure to remember to save your file in RTF format before compiling. Otherwise the compiler quickly chokes on this file.</p> |
|--|

The Viewer *Authoring Guide* tells you how to configure Word to eliminate the File Conversion dialog box. However, this dialog box can help you detect problems when a different format is shown, or the dialog box does not appear at all.

Standard hot spots appear in the compiled application as green underlined text. You can change this by using Topic Editor. With the second element (the Jump to command) selected, you can choose Hot Spot Formatting of None or Underlined and Colored. Choosing None makes your text appear in the color this text was formatted with. Use Character from the Format menu if you want to select a different color. You can't have the underlining, however. If you use a nonstandard appearance, be *certain* that your users understand. Be consistent!

## Create Additional Topics?

Complexity: EASY

### Problem

Now I want to create the topics I referenced in Contents. I also want users to be able to jump from one topic to the next (or previous) by using the Browse buttons.

### Technique

You create these topics just as you did the Contents topic, by using Topic Editor. You define the browse sequence similarly.

The files that are used or created in this How-To may be found in the VIEWERHT\HOWTOS\CHAP3\CHAP3\_3 directory on the enclosed CD-ROM disk. You build on the files created in the previous sections of this chapter, located on your hard disk in subdirectory VIEWERHT\CHAP3.

### Steps

1. Usually you prepare your directories and files next, but in this chapter continue to use the the directories and files from the previous sections.
2. Start Project Editor, and open your project file, \VIEWERHT\CHAP3\CHAP3.MVP.

Next update the document:

3. Double-click on the line naming your text file: TEXT\CHAP3.RTF.
4. Project Editor starts Word for Windows.
5. Word displays its Convert File dialog box, showing that the selected file is in Rich Text Format (RTF). Click on the OK button.
6. Your document appears, with the beginning of the Contents topic at the top of the window.
7. Make sure the paragraph symbol at the right-hand end of the Word ruler is depressed. This causes all hidden text, blanks, and control codes to be displayed.
8. Position the insertion point at the end of the document. You can insert new topics anywhere, but it's best to keep them in a logical sequence. That makes it easier to locate a particular topic, and it often places similar information close by, which can make it easier to write your text.

Now create the first topic, How Is a Viewer Application Created?

9. Use the hotkey you defined to bring up Topic Editor.
10. You want the first entry, Topic (page break and # footnote). Because it is already selected, click on OK.
11. Enter the context string you selected for this topic in the previous How-To, ctx\_chap\_2, and click on OK.
12. Bring up Topic Editor with your hotkey again, without moving the insertion point.

13. Select Topic title (\$ footnote). Click on OK.
14. Enter the title, *Designing a Viewer Application* and click on OK.
15. Bring up Topic Editor with your hotkey once again, without moving the insertion point.
16. Select Topic groups (+ footnote). Click on OK to display the dialog box.
17. Enter the word *main* as the Topic Browse Sequence.
18. Enter 010 as the Browse Sequence Number.
19. Leave the Topic Groups blank. The completed dialog box should look like Figure 3–13.

```
{ewc vwrht2, TsTextButton, "Figure
3½13"[Macro=JI('viewerht.mvb>SecWin', `fig3_13')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

20. Click on OK to return to Word.
21. Without moving the insertion point, press **[CTRL]–[SPACE]** to restore the normal format, select a larger font size, and click on the boldface button.
22. Type the topic heading, *Designing a Viewer Application*
23. Press **[ENTER]** then select Normal style to return to the standard format.
24. Drop down a few more lines.
25. Go to your Contents topic and highlight the hot spots under Chapter 2. Choose Copy from the Edit menu to copy this text to the Clipboard.
26. Return to the bottom of your new topic and insert the text you just copied by pressing **[SHIFT]–[INS]**.
27. Repeat these steps for the first topic under Chapter 2, Introduction. Use *chap2* as the Topic Browse Sequence, with Browse Sequence Number 010. Enter a couple of lines of suitable text under the topic instead of copying hot spots from the Contents.
28. Repeat the preceding steps for each of the remaining topics within Chapter 2. Increase the Browse Sequence Number by 10 for each topic. Notice that the values you used in the previous Browse definitions are displayed, so you don't have to reenter values that don't change. Be sure to add some appropriate text for each topic. The text for the topic titled *What Files Can Be Used?* should include a list of the supported file types. How–To 3.4 expects the term WMF to be used in that topic.
29. Repeat the same sequence of steps for Chapter 3. In the chapter–heading topic, use *main* as the Topic Browse Sequence, and 020 as the Browse Sequence Number. Copy the appropriate hot spots from the Contents topic as you did for Chapter 2.  
 In the remaining topics, use *chap3* as the Topic Browse Sequence, and use Browse Sequence Numbers starting with 010 and increasing by 10 for each topic. Enter a couple of lines of text in each as you did for Chapter 2.

Your document should look like Figure 3–14 after making these changes.

```
{ewc vwrht2, TsTextButton, "Figure  
3i½14"[Macro=JI('viewerht.mvb>SecWin', `fig3_14')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

30. Choose Save As from the File menu.
31. Select Rich Text Format (\*.RTF) under Save File as Type.
32. Click on OK.
33. Choose Exit from the File menu or double-click on the Windows control menu box to exit Word.
34. When the message *Do you want to save changes to CHAP3.RTF?* appears, you *must* click on No.

Next define the Browse groups. It actually is best to define them before creating the topics, because they were planned in advance.

35. You should be back in Project Editor. If not, switch to it now.
36. Choose Groups from the Section menu.
37. Click on New to define a new group.
38. Enter the name of the first group as main.
39. Enter a title for the group, such as Chapter Headings.
40. Leave the Searchable box checked.
41. Repeat steps 37 through 40 for groups chap2 and chap3. Figure 3–15 shows the dialog box after defining the last group.

```
{ewc vwrht2, TsTextButton, "Figure  
3i½15"[Macro=JI('viewerht.mvb>SecWin', `fig3_15')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

42. Click on OK.
43. Save your project file.
44. Now you can compile the application by choosing Build from the Compile menu.
45. There shouldn't be any errors this time. If there are, you can examine them after the compile is finished by choosing Display Error Log from the Compile menu.
46. Correct any errors and recompile until there are no errors reported. You may have to repeat this process several times.
47. Now test your application by choosing Run Viewer on CHAP3.MVB from the File menu.
48. When you click on a hot spot in the Contents, it should display the requested topic.
49. Click on a hot spot for a chapter title. It should display a topic with hot spots listing the sections within that chapter.
50. The Browse buttons should be grayed out in the Contents topic.
51. The Browse>> button should be active in the Chapter 2 topic (Designing a Viewer Application). Click on it—the Chapter 3 topic should be



displayed.

52. The Browse<< button should be active in the Chapter 3 topic. Click on it—the Chapter 2 topic should be displayed.
53. Within the topics for each chapter, the Browse>> button should be active in the first topic, the Browse<< button should be active in the last topic, and both Browse buttons should be active in the topics in between. Click on them. The >> button should always display the next topic in the chapter, and the << button should display the previous topic.
54. Try using the Go Back and History buttons. See what they do.
55. Click on the Search button. You should see a window like Figure 3–16.

```
{ewc vwrht2, TsTextButton, "Figure  
3½16"[Macro=JI('viewerht.mvb>SecWin', `fig3_16')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

56. Notice the Topic Groups listed. Look familiar? These are the groups you created for the Browse sequences. Enter a common word such as Contents and click on OK. Try it out.

### How It Works

Most of the steps in this example were the same as when you created the Contents topic. Concentrate on the new operations.

The first new operation was the definition of the Topic Groups entries. These set up the Browse sequences—which browse buttons are active and which topics they display. The name used in the topic browse sequence field defines a group of topics, all using the same browse sequence, that the user can browse through. The topics are displayed in the order of the browse sequence numbers. The << button is inactive in the first topic, because there is no previous topic to display. Similarly, the >> button is inactive in the last topic because there is no next topic.

The topic groups appear in the Search dialog because you leave the Searchable option on when you defined the groups. If you turn that option off, they only serve as Browse groups. The Search function is explored in great detail in Chapter 11.

The History button lists the titles of the topics that you recently displayed, in reverse order. These are the titles you defined through Topic Editor. Did your list show an Introduction topic? Could you tell which one it was? As you can see, these titles must be unique to avoid confusing the user.

### Comment

If you had defined the topic browse sequence names in Project Editor *before* you used them in the document, you could have selected the desired name from the pull-down list in the Topic Editor's dialog box. That would have made the definitions faster and less subject to error by eliminating some typing. Once again, planning and preparation make authoring faster and easier!

Browse Sequence Numbers should always be entered in increments of at least 10. This allows room for inserting new topics between existing ones. It

also allows room for changing the sequence of some topics without having to edit all of them.

The values associated with all of the topic definitions you enter (context string, title, and browse sequence) are stored as footnotes. Hot spots and most other commands are stored within the standard text. They use character formatting, paragraph formatting, and curly braces—{}—to distinguish commands from normal text.

Would you like to be able to look at the footnotes and see the values? Just select Footnotes from Word's View menu and a window opens showing them. Word keeps the two windows synchronized; if you move the insertion point within one window, the other will move with it.

You were able to copy the hot spot text from one topic to another because it is just specially formatted text. There is nothing magical about these entries. You could also copy a footnote definition by copying the footnote code. The values associated with that would be copied, too.

Context strings can be defined in the body of a topic, as well as in the heading. A hot spot that uses a context string located within a topic displays the text starting with the beginning of the paragraph where the context string is defined, rather than the beginning of the topic. If you create a hot spot near the end of a topic, be sure to add enough blank lines at the end of the topic to fill the remainder of the pane.

While compiling you may get a compiler message saying  
{vfld2305858952132296716} Warning 6181 {vfld2305857852620668928}:  
*Specified word breaker unable to process word.* This is due to a bug in Viewer, and you may ignore it.  
{vfld137438953484} Warning 6182 {vfld280933810831360} has also been reported similarly. These warnings usually seem to be caused by temporary files left over from previous compiles—they go away if you recompile the same files.

## Create a {vflid137438953482}Popup{vflid3800470531342336} Topic?

Complexity: EASY

### Problem

I've seen Windows Help display information in a popup window when I click on a hotspot. I want to do that in my Viewer application to help define some terms that may be unfamiliar to the user.

### Technique

A popup displays the contents of a topic that is defined much like any other. The hot spot displays it as a popup rather than in the full window. You use Topic Editor to prepare both the topic and the hot spot.

You continue to use the application you've been building in this chapter. You use the popup to define the file type WMF in the second section of Chapter 2. If you didn't use that term in your text, you add it here.

The files that are used or created in this How-To may be found in the VIEWERHT\HOWTOS\CHAP3\CHAP3\_4 directory on the enclosed CD-ROM disk. This How-To builds on the files created in the previous sections of this chapter, located on your hard disk in subdirectory VIEWERHT\CHAP3.

### Steps

First prepare your directories and files:

1. Usually you would prepare your directories and files next, but for this chapter continue to use the the directories and files from the previous sections.
2. Start Project Editor, and open your project file, \VIEWERHT\CHAP3\CHAP3.MVP.

Next update the document:

3. Double-click on the line naming your text file: TEXT\CHAP3.RTF.
4. Project Editor starts Word for Windows.
5. Word displays its Convert File dialog box, showing that the selected file is in Rich Text Format (RTF). Click on the OK button.
6. Your document appears, with the beginning of the Contents topic at the top of the window.
7. Make sure the paragraph symbol at the right-hand end of the Word ruler is depressed. This causes all hidden text, blanks, and control codes to be displayed. (I'm repeating this in every How-To in this chapter to help make it automatic. I won't mention it again in the other chapters.)
8. Position the insertion point at the end of topic 4, titled Viewer's Files. You can insert new topics anywhere, but it's best to keep them in a logical sequence. That makes it easier to locate a particular topic, and it often places similar information close by, which can make it easier to write text.
9. Use the hotkey you defined to bring up Topic Editor.

10. You want the first entry, Topic (Page break and # footnote). Since it is already selected, click on OK.
11. Enter the context string for this popup, `ctx_pop_wmf`, and click on OK.
12. Don't define a topic title or browse sequence for this topic. These options aren't used for popup topics, since these topics are only displayed through hot spots.
13. Enter the text you want displayed in the popup window. This should explain what the term "WMF" means. For example:  
This refers to the Windows Meta File format. It contains instructions to draw lines and circles, rather than the contents of the resulting pixels. WMFs are very small files that display with high precision at all resolutions.
14. Display the previous topic (Viewer's Files), and select the term WMF in your text.
15. Bring up Topic Editor with your hotkey.
16. You want the first entry, Hot spot (text). It is already selected, so click on OK.
17. While Hot Spot is selected on the left side of the window, click on Popup under the Hot Spot Type on the right side.
18. Select the Jump to... element.
19. Enter the context string for the popup topic, `ctx_pop_wmf`. Click on OK. Your popup should look like Figure 3–17.

```
{ewc vwrht2, TsTextButton, "Figure
3i½17"[Macro=JI('viewerht.mvb>SecWin', `fig3_17')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

20. Notice that the displayed text has a single underline, rather than the double underline used for jumps.
21. Save your file and compile your application. If you've forgotten how to do this, reread How-To 3.3.
22. Test your application. You should see something like Figure 3–18.

```
{ewc vwrht2, TsTextButton, "Figure
3i½18"[Macro=JI('viewerht.mvb>SecWin', `fig3_18')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

23. Notice the slightly different appearance of the hot spot. It is displayed with a dotted underline, rather than the solid underline you saw for jumps.
24. Resize your Viewer window smaller and click on the popup hot spot. The popup window may extend outside the Viewer window.

**Comment**

You cannot control the size, margins, or position of a popup window created

in this way. Viewer determines the size based on the amount of text to be displayed. The position is based on the location of the mouse pointer. How-To 5.4 shows how to use a custom popup pane within your window.

Popup windows are most commonly used to provide definitions of terms, as in this example. They can serve any purpose compatible with their temporary nature.

## **Create Searchable {vflid137438953482}Keywords{vflid3800470531342336}?**

Complexity: EASY

### **Problem**

I want to define specific terms the user could use to search for information. This must serve in addition to the existing full-text search.

### **Technique**

You use Topic Editor to define keywords in your topics. These can consist of more than one word, and do not have to be terms that are actually used in the document. The user can locate topics using the keywords by clicking on the Index button, just as the Search button is used for the full-text search.

Although keywords are usually defined at the very beginning of a topic, you also create some in the middle to see what happens.

The files that are used or created in this section may be found in the VIEWERHT\HOWTOS\CHAP3\CHAP3\_5 directory on the enclosed CD-ROM disk. You build on the files created in the previous sections of this chapter, located on your hard disk in subdirectory VIEWERHT\CHAP3.

### **Steps**

First prepare your directories and files:

1. Usually you would prepare your directories and files next, but in this chapter continue to use the the directories and files from the previous sections.
2. Start Project Editor, and open your project file, \VIEWERHT\CHAP3\CHAP3.MVP.

Next you can update the document:

3. Double-click on the line naming your text file: TEXT\CHAP3.RTF.
4. Project Editor starts Word for Windows.
5. Word displays its CONVERT FILE dialog box, showing that the selected file is in Rich Text Format (RTF). Click on the OK button.
6. Your document appears, with the beginning of the Contents topic at the top of the window.
7. Make sure the paragraph symbol at the right-hand end of the Word ruler is depressed. This causes all hidden text, blanks, and control codes to be displayed.
8. Display the second topic, Designing a Viewer Application.
9. Position the insertion point at the beginning of the topic heading, just after the last footnote code.
10. Use your hotkey to bring up Topic Editor.
11. Select Keywords (K footnote) and click on OK to display the dialog box.
12. Leave the Keyword Index unchanged.

13. Enter one or more keywords as shown in Figure 3–19.

```
{ewc vwrht2, TsTextButton, "Figure
3i½19"[Macro=JI('viewerht.mvb>SecWin', `fig3_19')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

14. Click on OK to return to Word.
15. Define keywords in the heading of other topics similarly. Define Introduction as a keyword in both Introduction topics.
16. Save the file and compile your application as usual. Next, test your changes.
17. Click on the Index button. You should see a list of all the keywords you defined, with the number of topics where each was defined, as shown in Figure 3–20.

```
{ewc vwrht2, TsTextButton, "Figure
3i½20"[Macro=JI('viewerht.mvb>SecWin', `fig3_20')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

18. Select a keyword that was defined once, and click on OK. Viewer displays the topic where it was defined.
19. Click on the Index button again, select a keyword that is defined more than once, and click on OK. There should be at least one, since you defined Introduction twice.
20. Viewer displays a list of the topic titles where this keyword is defined. Select one and click on Go To. The selected topic is displayed.

### **How It Works**

The keywords are stored in your document as footnotes, using the letter *K* as the footnote code. The compiler assembles the footnotes into an index similar to the index used by the Search function.

### **Comment**

Keywords don't have to be terms that are actually used in your text. They can be common terms that the user would understand, or terms the user might use to describe something in your application.

These keywords work exactly like the ones in the Windows Help system. WinHelp allows multiple keyword indexes to be defined, using different footnote code letters. The WinHelp secondary indexes can only be accessed by a program calling Help. By contrast, Viewer lets the author define secondary indexes through Topic Editor dialog box that can be easily used by your user. Try this on your own. The secondary index groups must be defined through Project Editor's Sections menu before you use them in your document. You should have no problem finding the right menu item and completing the information in the dialog box.





## Keep the Table of Contents Visible?

Complexity: INTERMEDIATE

### Problem

I want to keep my table of contents visible to the users while they look at the topics, to make it easier for them to move around in my application.

### Technique

You use Project Editor to define two panes in your window, with the Contents topic in one and all other topics in the other. For this to work properly, you have to create one additional topic that appears in the regular pane, next to the Contents, when the file is loaded and whenever the user asks to display the Contents.

The files that are used or created in this section may be found in the VIEWERHT\HOWTOS\CHAP3\CHAP3\_6 directory on the enclosed CD-ROM disk. You build on the files created in the previous sections of this chapter, located on your hard disk in subdirectory VIEWERHT\CHAP3.

This process goes through many different dialog boxes and can be confusing. This section includes even more pictures than usual to help make it clearer.

### Steps

1. Usually you would prepare your directories and files next, but in this chapter continue to use the the directories and files from the previous sections.
2. Start Project Editor, and open your project file, \VIEWERHT\CHAP3\CHAP3.MVP.

Next define the window panes:

3. Choose Window Definitions from the Section menu. The Window Definitions dialog box shown in Figure 3–21 is displayed.

```
{ewc vwrht2, TsTextButton, "Figure
3i½21"[Macro=JI('viewerht.mvb>SecWin', `fig3_21')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

4. Click on the Properties button to display the Window Properties dialog box.
5. Select the Use Default Color checkbox, and enter the Window Caption as Viewer Demo App. Figure 3–22 shows the completed dialog box.

```
{ewc vwrht2, TsTextButton, "Figure
3i½22"[Macro=JI('viewerht.mvb>SecWin', `fig3_22')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

6. Click on the Master Pane button to display the Master Pane Properties

dialog box.

7. Clear the Auto-Position check box. The Master pane cannot be resized to allow room for your new pane if this is selected. The dialog box should look like Figure 3-23.

```
{ewc vwrht2, TsTextButton, "Figure  
3-23"[Macro=JI('viewerht.mvb>SecWin', `fig3_23')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

8. Click on the Preview On button. Viewer displays your main window and master pane.
9. Click anywhere in the master pane, and locate the handle on the left edge. It's small and hard to see.
10. Drag the handle to the right until the master pane covers only the right half of the window. Figure 3-24 shows this window.

```
{ewc vwrht2, TsTextButton, "Figure  
3-24"[Macro=JI('viewerht.mvb>SecWin', `fig3_24')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

11. Close the preview window by double-clicking on the control menu box.
12. Choose the Panes file folder tab.
13. Click on the New button and then the Properties button. The Pane Properties dialog box is displayed.
14. Enter the pane name as **contents**.
15. In the Dismiss When box, select Title Is Closed. The dialog box should look like Figure 3-25.

```
{ewc vwrht2, TsTextButton, "Figure  
3-25"[Macro=JI('viewerht.mvb>SecWin', `fig3_25')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

16. Click on the Windows button to display the Pane Associations dialog box.
17. Under Show pane "contents" in which windows?, select Main.
18. Select the Show in Window check box. The dialog box should look like Figure 3-26.

```
{ewc vwrht2, TsTextButton, "Figure  
3-26"[Macro=JI('viewerht.mvb>SecWin', `fig3_26')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

19. Click on the Preview button to display the window. You see the new regular pane, labeled Contents, on the left and the master pane on the right.

20. Click on the regular pane to display the handles, and drag them until the Contents pane covers the left side of the window as shown in Figure 3–27. Viewer shows both the scrolling and non–scrolling regions in the master pane, which looks like another pane. Don't be confused by this.

```
{ewc vwrht2, TsTextButton, "Figure
3i½27"[Macro=JI('viewerht.mvb>SecWin', `fig3_27')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

21. Double–click on the control menu box to close the window.
22. Click on OK in the Window Definitions dialog box to return to Project Editor.
23. Choose Save from the File menu to save your changes.

Next, create the new topic and adjust the context strings:

24. Double–click on the text file name to start Word.
25. Press **[ENTER]** to create a blank line before the start of the Contents topic.
26. Use your hotkey to invoke Topic Editor.
27. Select the Topic entry and click on OK.
28. Enter the desired context string as `ctx_contents` and click on OK.
29. Use the hotkey to invoke Topic Editor again.
30. Select Topic–Entry Command and click on OK.
31. Click on Paste Command, and select the `{vfld137438953483}PaneID{vfld3940108508069888}` command as shown in Figure 3–28. This selects the command to be executed when the topic is entered. Click on OK.

```
{ewc vwrht2, TsTextButton, "Figure
3i½28"[Macro=JI('viewerht.mvb>SecWin', `fig3_28')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

32. Click on Edit Command to set the command parameters.
33. Enter the file name as ``CHAP3.MVB'`, window name as ``main'`, context string as ``ctx_contents_2'`, pane name as ``contents'`, and `PrintTabCopyOrder` as 0. This dialog box is shown in Figure 3–29.

```
{ewc vwrht2, TsTextButton, "Figure
3i½29"[Macro=JI('viewerht.mvb>SecWin', `fig3_29')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

34. Click on OK in each dialog box to return to Word.
35. Press **[CTRL]–[ENTER]** to create a hard page break between the topics.
36. Select the # footnote code at the beginning of the Contents topic, and activate Topic Editor. It will immediately show the context string dialog

- box with the current context string.
37. Change the context string to `ctx_contents_2` and click on OK.
  38. Save your file as an RTF file and exit Word.
  39. Compile your application, and test it. Your initial window should look like Figure 3–30.

```
{ewc vwrht2, TsTextButton, "Figure
3½30"[Macro=JI('viewerht.mvb>SecWin', `fig3_30')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

Viewer automatically displays the contents topic when the file is loaded. You previously defined this to be the topic with context string `ctx_contents`. Here, you gave the new topic this context string, and changed the former contents topic to context string `ctx_contents_2`.

You then defined the `PaneID` command to be executed whenever the Contents topic is displayed. This command specifies that the topic with context string `ctx_contents_2` is displayed in the contents pane in the main window.

The vertical line separating the topics is the border of the master pane. Notice how close the master pane text is to that line. Viewer does not provide any margin, to allow you the greatest possible control over the appearance of your application. You can create a margin by using left or right indent paragraph formatting in Word. Note that document format options are ignored.

Notice the border around the table of contents. Regular panes are automatically resized to fit the topic contents, which causes the border to be displayed close to the text. This would look better without a border. This resizing is also the reason why you checked the `Use Default Color` checkbox for the window. If you hadn't, a different color (the window color) might show around the regular pane. You usually should use the same background colors for the window and panes. The default color is the best choice, since it lets the user control the appearance. Try removing the border and selecting different colors to see these effects.

### Comment

There can only be one master pane, with as many regular panes as desired.

Only the master pane can have scroll bars. They display automatically if the topic is too large to fit within the pane. The topics displayed in any regular panes are clipped if they are too large to fit. Regular panes automatically shrink to fit smaller topics—keep this in mind if you display a border around a regular pane. It just surrounds your topic.

The largest picture that can be displayed in the master pane without scroll bars, with no border specified, is the width of the pane and one pixel less than the height of the pane. If you specify a one-pixel border, the limits are two pixels less than the width and three pixels less than the height.

All topics are displayed in the master pane unless forced into a regular pane through a PaneID command.

Panes can be permanent, as in this example. They can also be used for brief periods for particular purposes. You can use one pane or many. It is just a way of organizing your information in the window—much like you might use a table to organize material within a topic.

Panes can be used to provide a particular user interface. They can contain any buttons, hot spot pictures or hot spot text you want. You could use more than one pane this way—placing buttons on both sides of the master pane, for example.

Topic entry and group entry commands are only executed for topics displayed in the master pane. If topics with such commands are displayed in regular panes, the commands are ignored.

The PrintTabCopyOrder field determines the sequence in which panes are printed or copied. A value of zero prevents the pane from being printed or copied.

## 3.7 Tips and Tricks

### Using Word Features

- ⇒ Define standard styles for your common headings and text, just as you do when creating a document to be printed. Your Normal style defined in Word should match the text in the body of your topics. Your Normal style should also have Use As Default selected. This prevents Word from creating unnecessary character-formatting commands in your RTF file, thus reducing the file size.
- ⇒ Use Auto color in all your character formats. This causes your text to be displayed using the colors selected in the Windows Control Panel. If you force black text, it is unreadable on systems that are configured for light text on a dark background.
- ⇒ Format your headings using Word's standard heading 1, heading 2, and heading 3 styles for topic headings and subheads, and use the Outline View to see all your headings. This can be very handy for locating a particular topic or section.
- ⇒ If you perform some functions frequently while in Word, other than operations involving Topic Editor, you can make them easier by creating macros. A macro, in its simplest form, is just the recording of a series of keystrokes or operations. The same keystrokes and operations can be repeated by executing the macro. Simple macros can be created by choosing Record Macro from Word's Tools menu, entering the desired macro name in the dialog box that appears, entering the keystrokes and mouse actions to be repeated, and finally choosing Stop Recorder from the Tools menu. Macros can be executed by selecting Macro from the Tools menu, selecting the macro name, and clicking the Run button. The creation and use of macros is explained in the Word documentation. The Viewer documentation includes instructions for creating a macro to save your document as an RTF file by selecting a single menu item.
- ⇒ Some operations are easier if you use Word's keyboard shortcuts, especially when you are in the middle of typing. For example, **[CTRL]-[SPACE]** resets temporary character format changes, and **[CTRL]-[B]** toggles boldface. These are described in Word's documentation. Standard Windows functions, such as using Tab and arrow keys to move between controls in a dialog box, are also useful.
- ⇒ If you use Word to create printed documents as well as Viewer files, you may want different standard styles or macros. In this case you should create a document template (DOT file) for the less common environment, and define the appropriate styles and macros as local to that template. Templates can be applied to Viewer documents by choosing Template from Word's File menu. The creation and use of templates is explained in the Word documentation.
- ⇒ The Viewer documentation suggests setting a global option in Word to suppress the initial File Conversion dialog box. Personally, I prefer to see the dialog box. Its absence is the best way to find that you accidentally saved your document in Word's DOC format using the RTF file extension. This can be easily done by clicking on the Yes button in the Save dialog box that appears when exiting Word. This dialog box is also

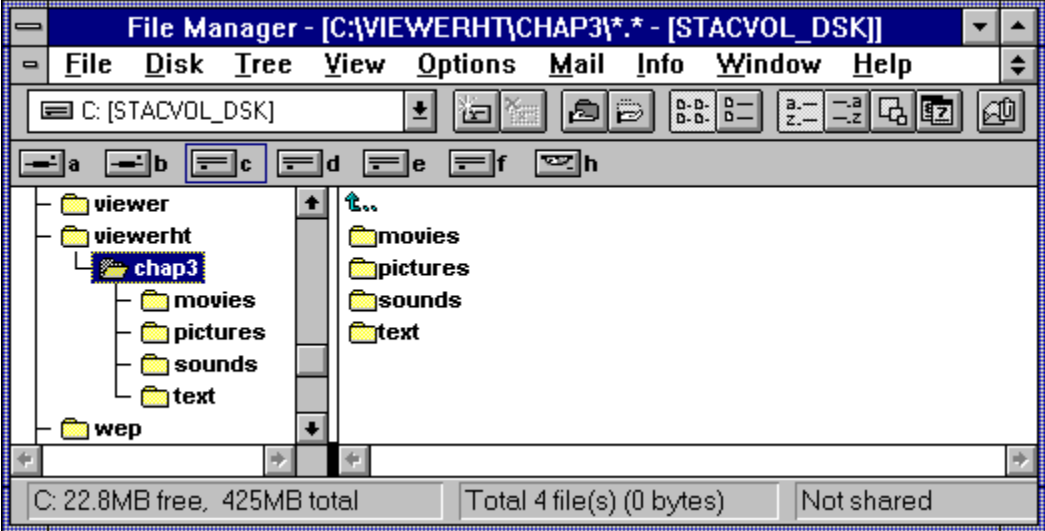
useful if you use Word for other functions besides creating Viewer files. If desired, you can create a macro in your Viewer template file to suppress this dialog box. Just use Word's macro recorder function as described above while following the instructions in the Viewer documentation. If you do this, you also need to create a similar macro in all of your other templates to restore your normal operation. If you name these macros AutoOpen in each template, they execute automatically when you open a document.

### **Editing Topic Commands**

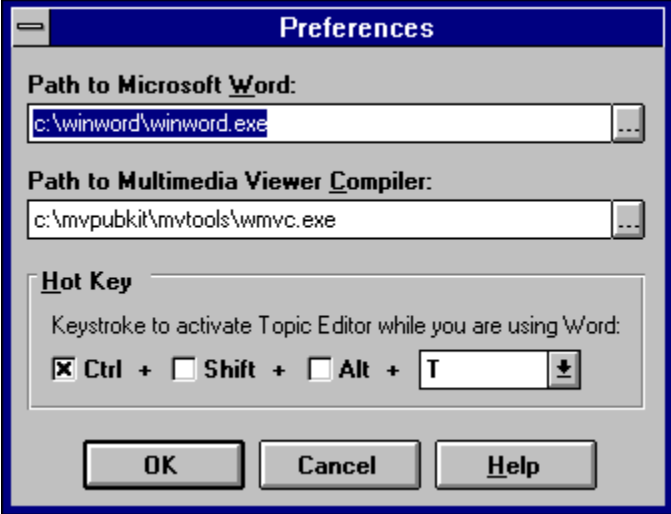
- ⇒ If you select the text of a command you created previously and call up Topic Editor, it picks up the selected command and allows you to edit it. This is generally much easier than trying to edit the command text in your document.
- ⇒ If you need similar Viewer commands in several locations within a topic, copy the command text using standard Word operations, then edit them as necessary.
- ⇒ Editing an existing command does not work if the command is in a table cell. This will be fixed in a future release of Viewer.
- ⇒ If you highlight multiple topics—even an entire topic—Topic Editor displays each command found in the selected text and lets you edit any or all of them together. This can be a fantastic time saver! This is an undocumented feature of Viewer 2.0. Hopefully it will be documented, and thus can be counted on to remain supported, in the next release.

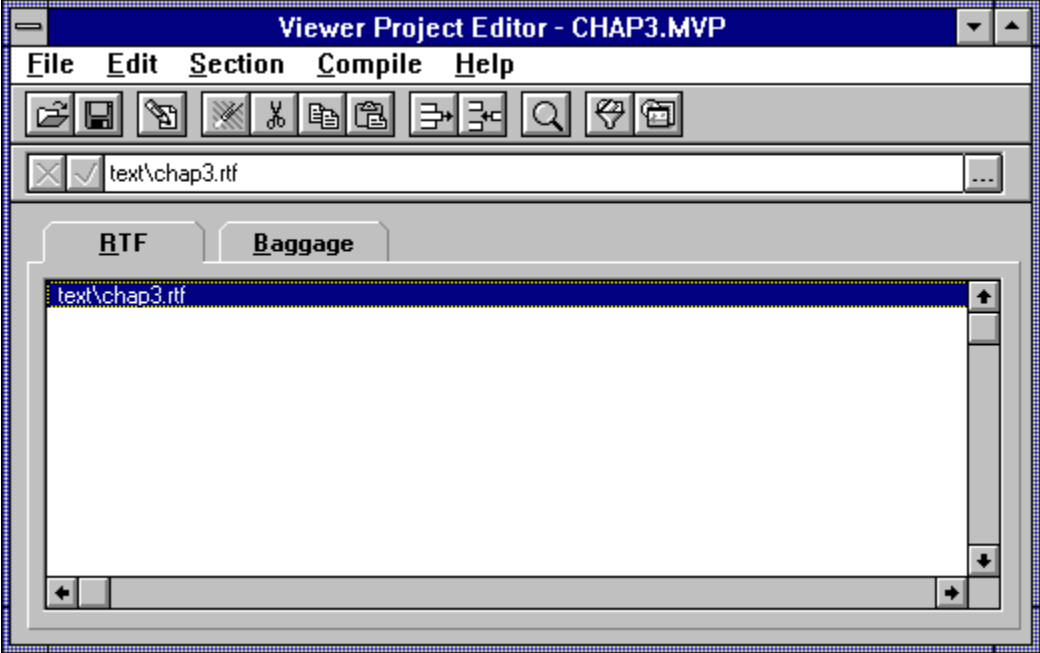
### **Miscellaneous**

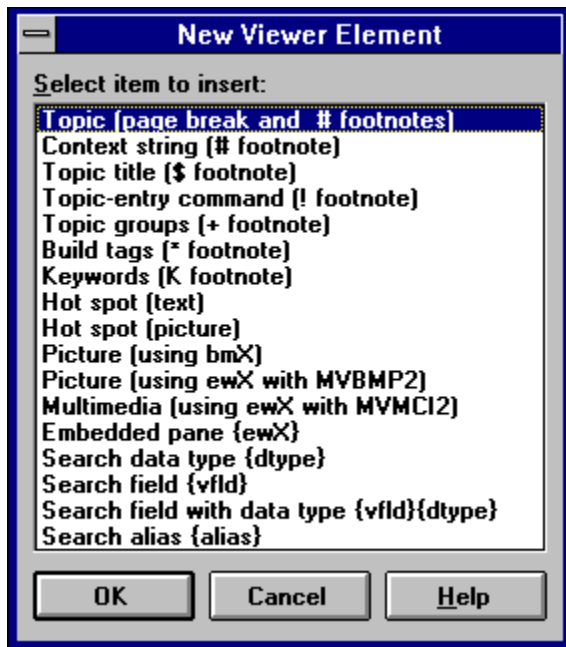
- ⇒ The Topic Editor function to create a new topic only lets you define the context string. The developers are working on allowing you to define optional topic titles and keywords in the same operation. This will probably be included in the next release.
- ⇒ When entering options in Topic Editor, be careful to leave the single quote marks that are included in many entries by selecting only the text between those marks. If you accidentally delete them, the left quote (') is on the key with the Tilde (~), above the Tab, on most keyboards. The right quote (') is usually on the key with the double quote mark ("). The Viewer developers have been asked to make this process simpler.

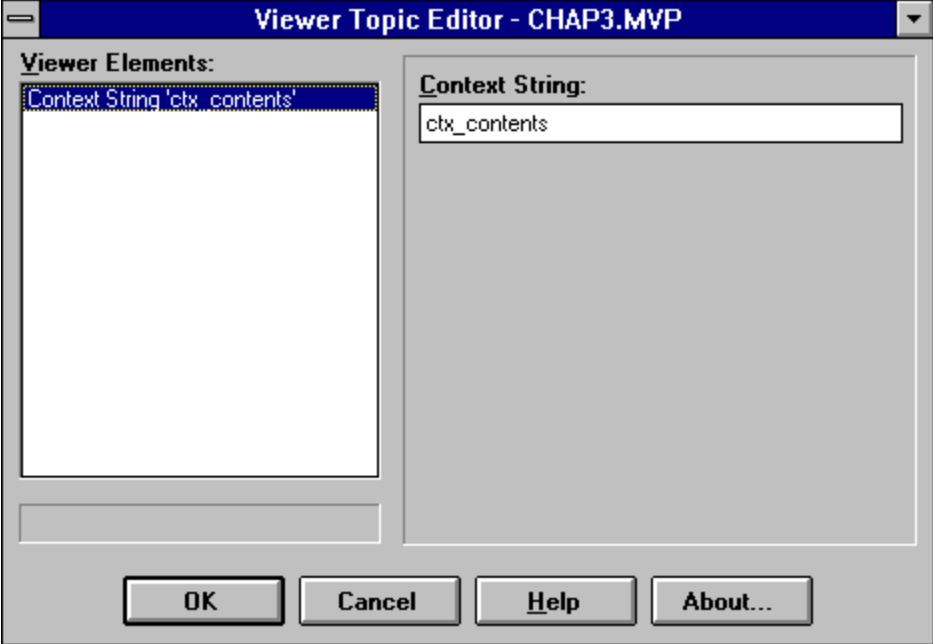


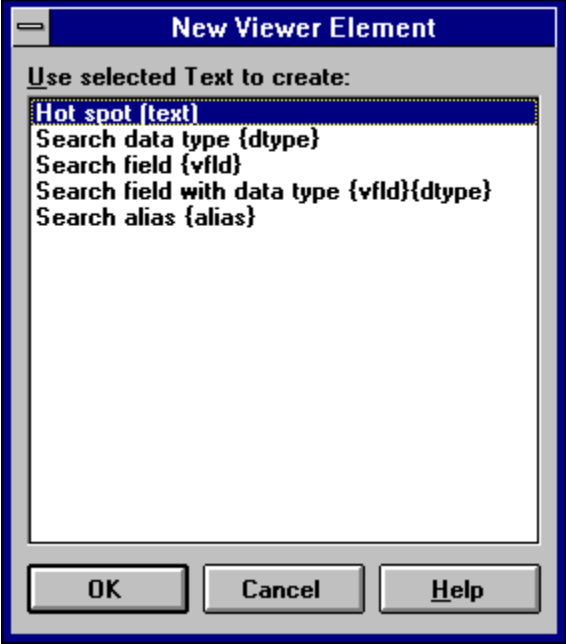


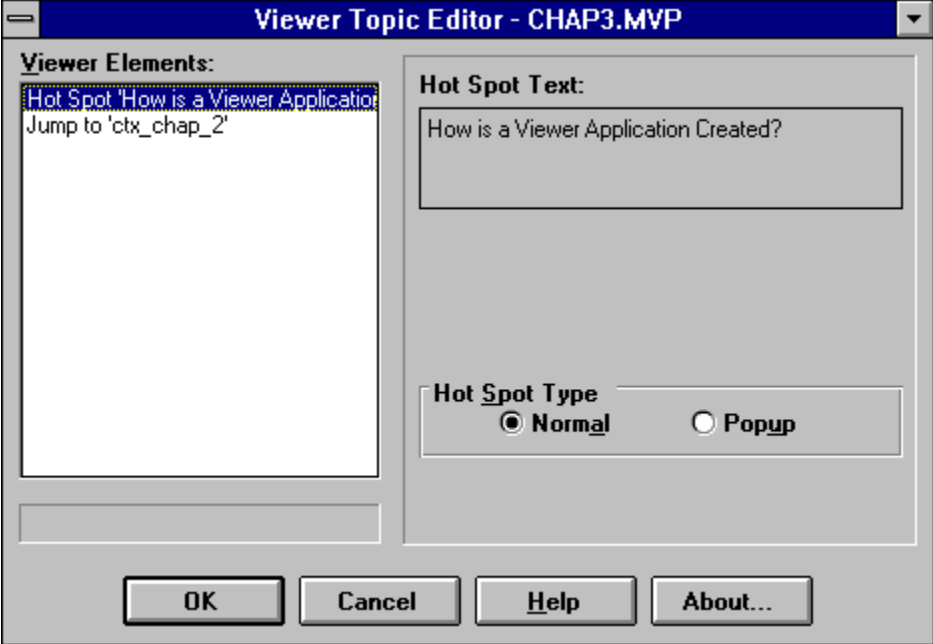


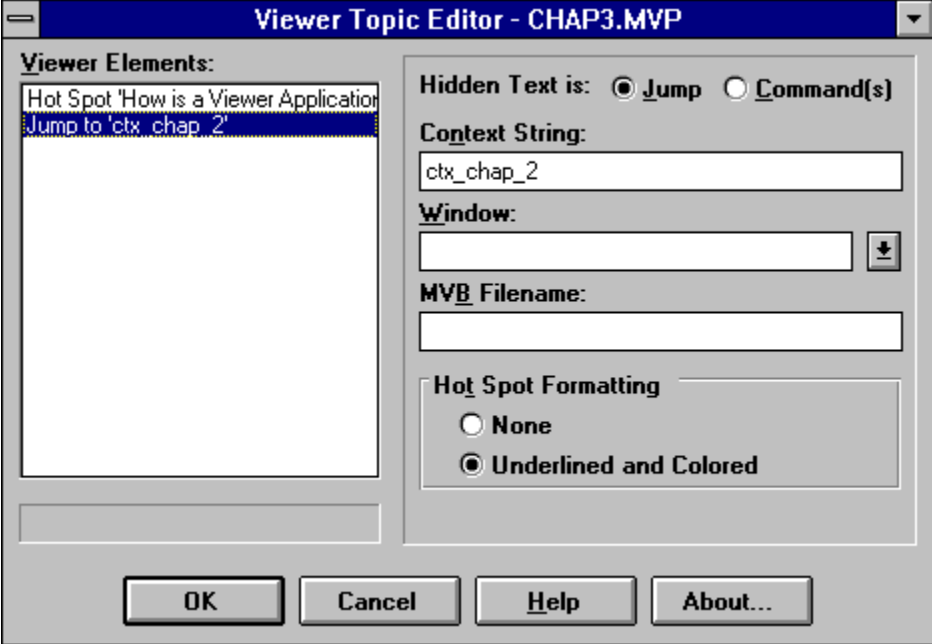












## ## Contents¶

¶

### Designing a Viewer Applicationctx\_chap\_2¶

- Introductionctx\_2\_intro¶
- How Should I Design the Application?ctx\_2\_design¶
- Viewer's Filesctx\_2\_files¶
- What Are the Authoring Tools?ctx\_2\_authoring¶
- What Are the Support Tools?ctx\_2\_support¶

### Creating a Simple Applicationctx\_chap\_3¶

- Introductionctx\_3\_intro¶
- Start a New Projectctx\_3\_project¶
- Create a Contents Topicctx\_3\_contents¶
- Create Additional Topicsctx\_3\_add¶
- Create a Popup Topicctx\_3\_popup¶
- Create Searchable Keywordsctx\_3\_keywords¶
- Keep the Table of Contents Visiblectx\_3\_visible¶
- Tips and Tricksctx\_3\_tips¶



**[OPTIONS] - Title Options**

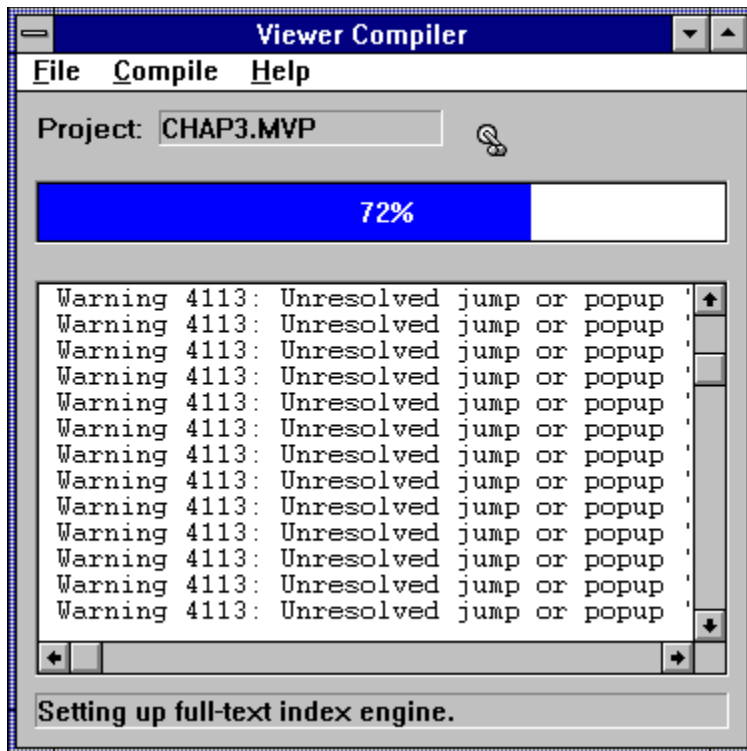
**C**lipboard Citation:

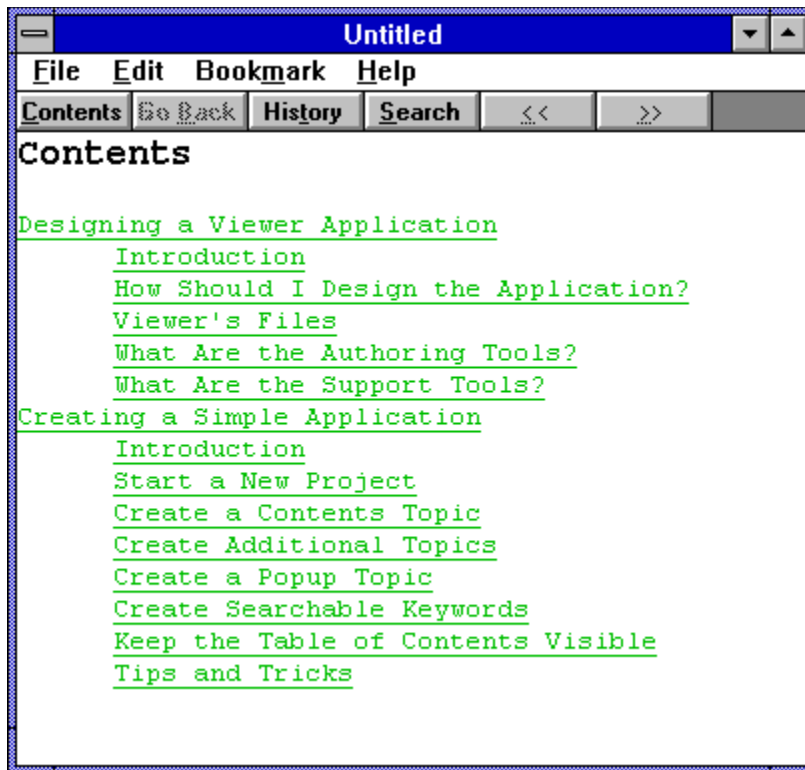
**C**ontents Topic:

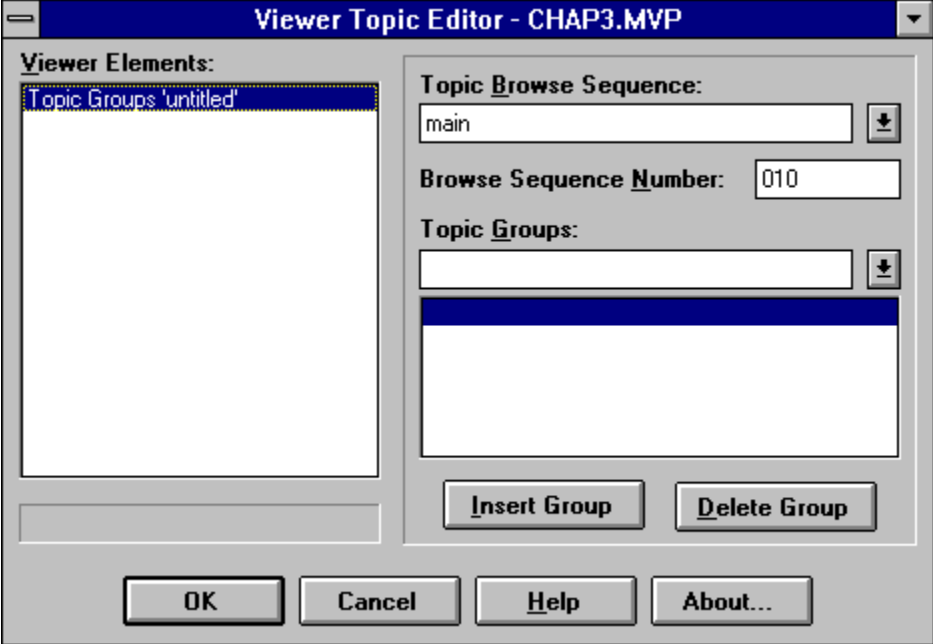
**C**opyright:

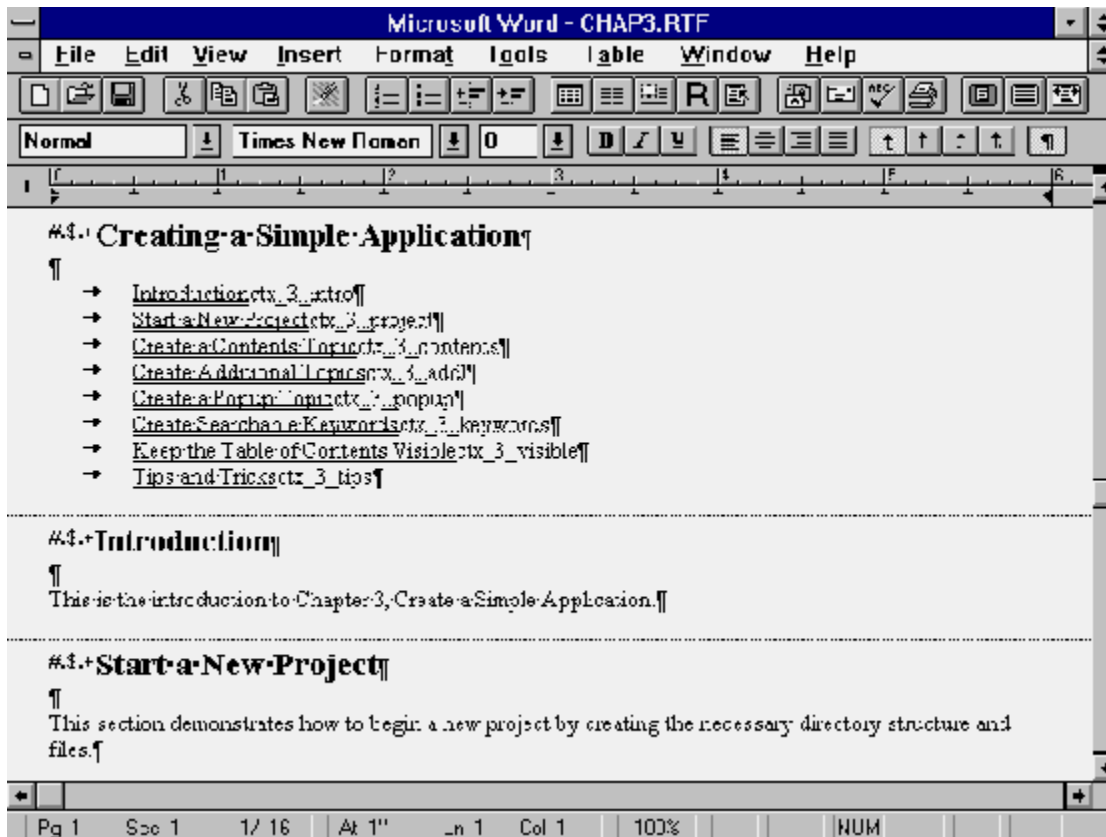
**I**con Filename:  ...

**T**itle:









[GROUPS] - Groups

**Topic Groups:**

- main
- chap2
- chap3**

**Name:** chap3  **Searchable**

**Title:** Chapter 3

**Comment:**

**Delete** **New** **Entry Script...** **Exit Script...** **OK** **Cancel** **Help**

**Search**

Search by Word

Search In:

- All Topic Groups
- Selected Topic Groups
- Current Topic Only

Topic Groups:

- Chapter Headings
- Chapter 2
- Chapter 3

OK

Cancel

Options...

Hints...

## ##-Viewer's-Files¶

¶

This section describes the files used by Viewer, the file formats that are supported, and other information related to Viewer files and disk usage.¶

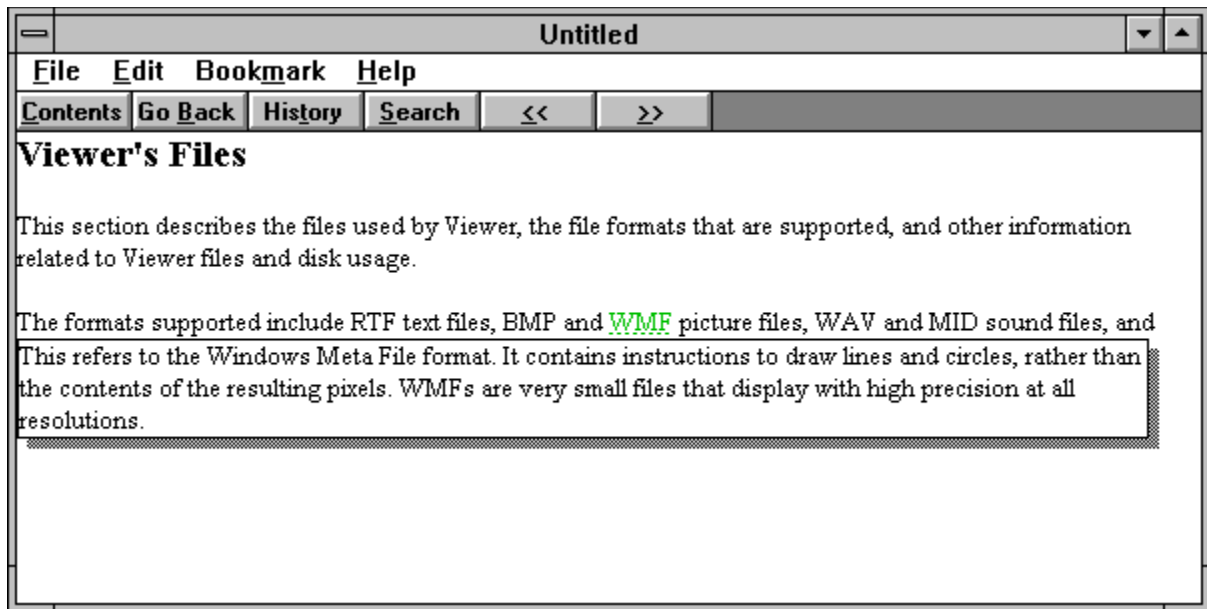
¶

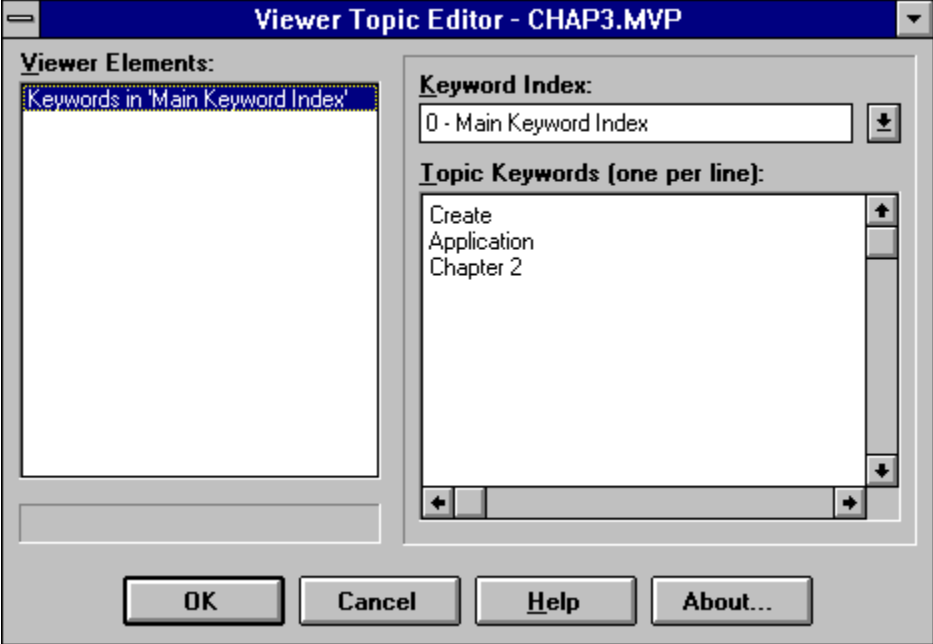
The formats supported include RTF text files, BMP and WMF (x\_300\_.wmf) picture files, WAV and MID sound files, and AVI movie files.¶

---

#This refers to the Windows Meta File format. It contains instructions to draw lines and circles, rather than the contents of the resulting pixels. WMFs are very small files that display with high precision at all resolutions.¶





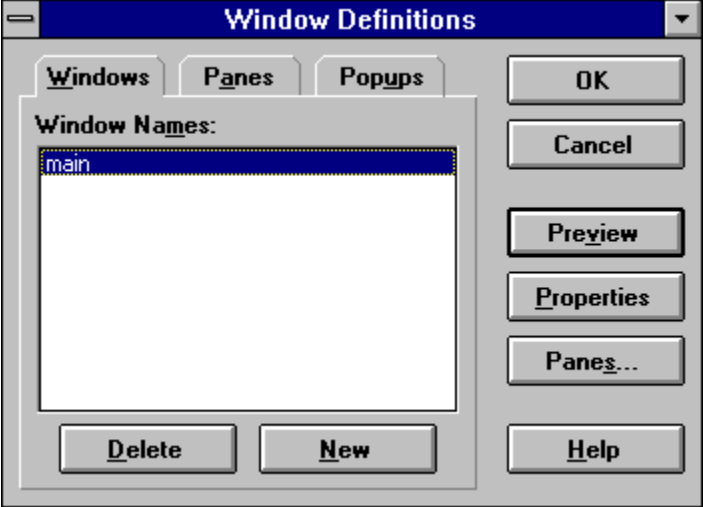


**Index**

Type a word, or select one from the list.

| Index list        | # of Topics |
|-------------------|-------------|
| <b>Additional</b> | <b>1</b> ↑  |
| Application       | 1           |
| Authoring         | 1           |
| AVI               | 1           |
| BMP               | 1           |
| Chapter 2         | 6           |
| Chapter 3         | 9           |
| characteristics   | 1           |
| Contents          | 1           |
| Create            | 4           |



**Window Properties**

**Window Name:**

**Top:**       **Height:**

**Left:**       **Width:**

**Auto-Position**

**Background Color:**  ...  **Use Default Color**

**Background Picture:**  ...

**Window Caption:**

**Initial State:**  ▾

**Preview Menu and Button Bar**

**Coordinate System:**  ...

**Master Pane Properties**

**Pane Name:** Master Pane

**Top:** 0 **Max Height:** 1023

**Left:** 470 **Max Width:** 553

**Auto-Position**

**Background Color:**    **Use Default Color**

**Border:** One-Pixel

**Non-Scrolling Region**

**Background Color:**    **Use Default Color**

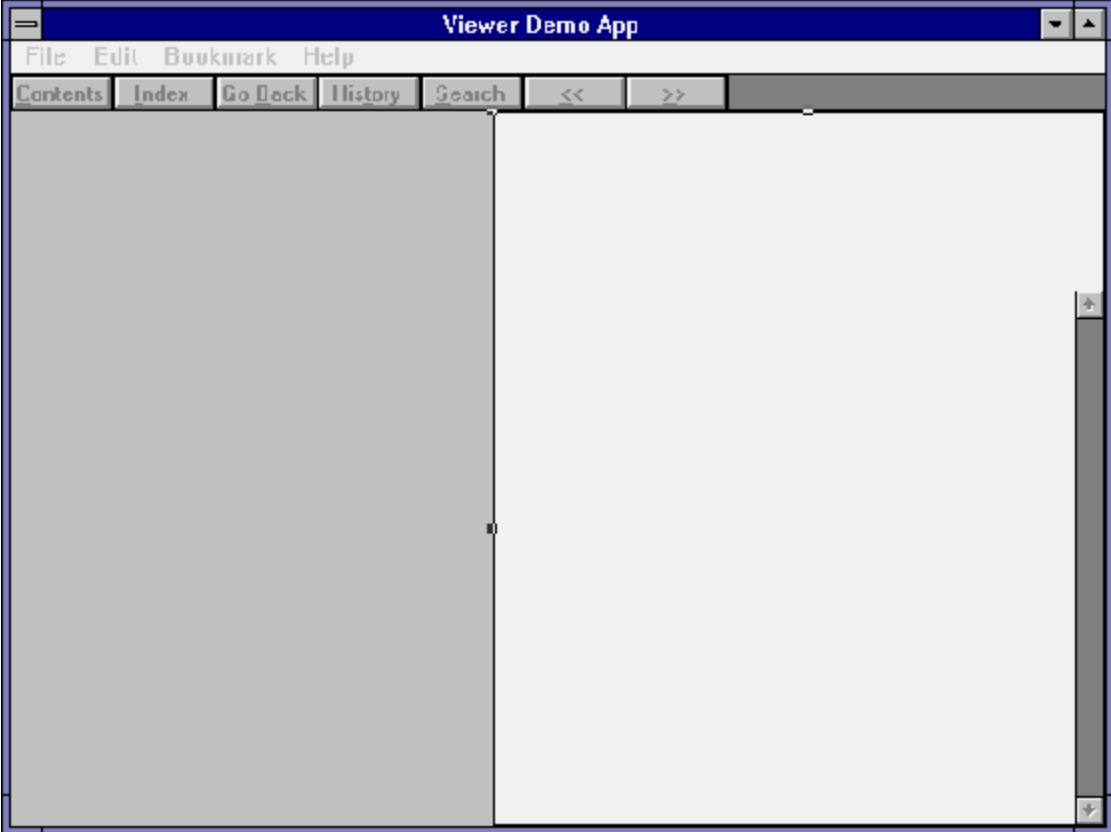
**Position:** Top

**Separator:** One-Pixel

**Minimum Margin**

**Horizontal:** 0 **Vertical:** 0

**Coordinate System:** Device-Independent (1024 x 1024)



**Pane Properties**

Pane Name: contents

Top: 0    Max Height: 878

Left: 0    Max Width: 470

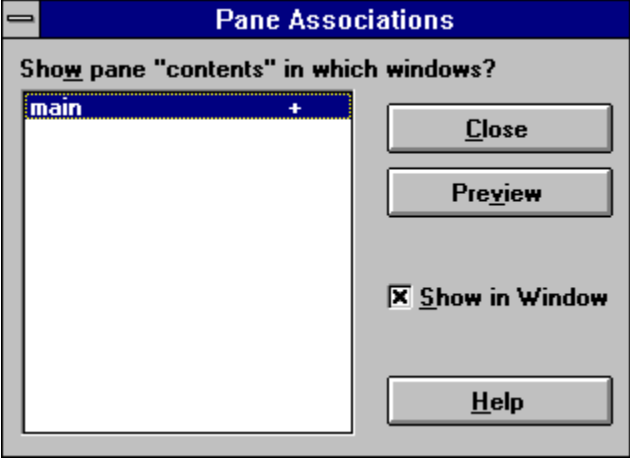
Background Color:  ...  Use Default Color

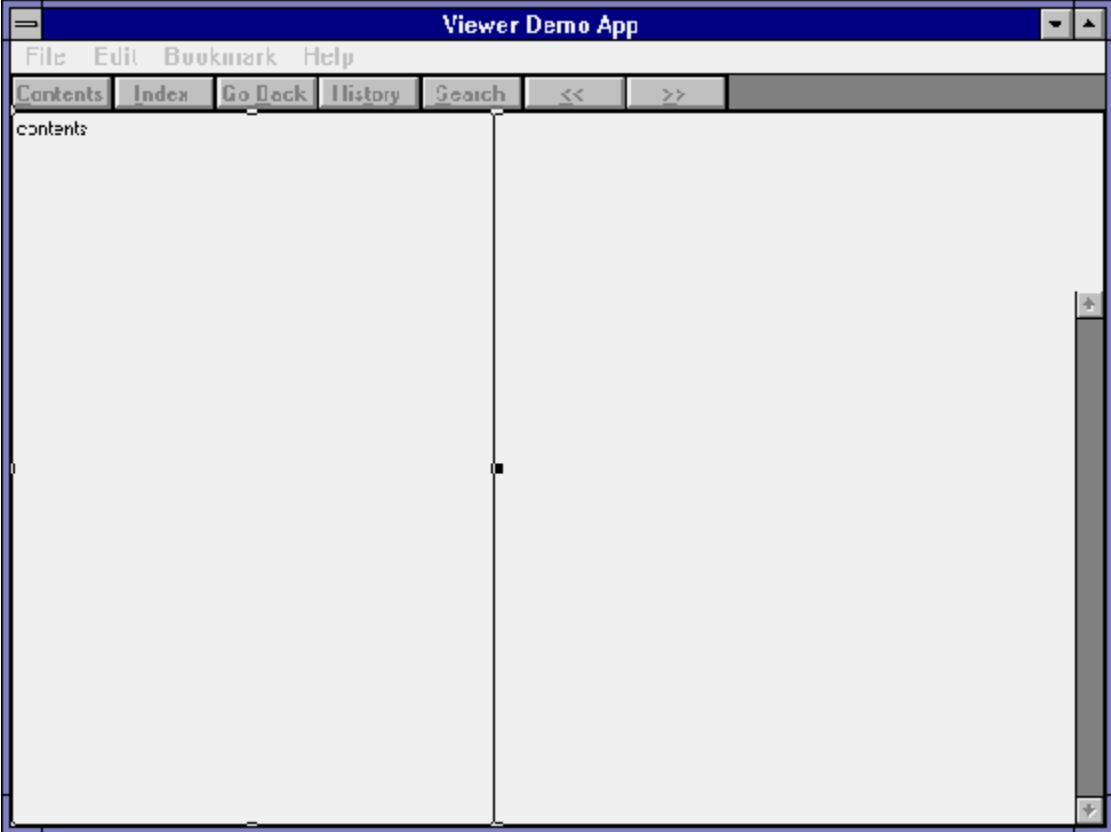
Border: One-Pixel

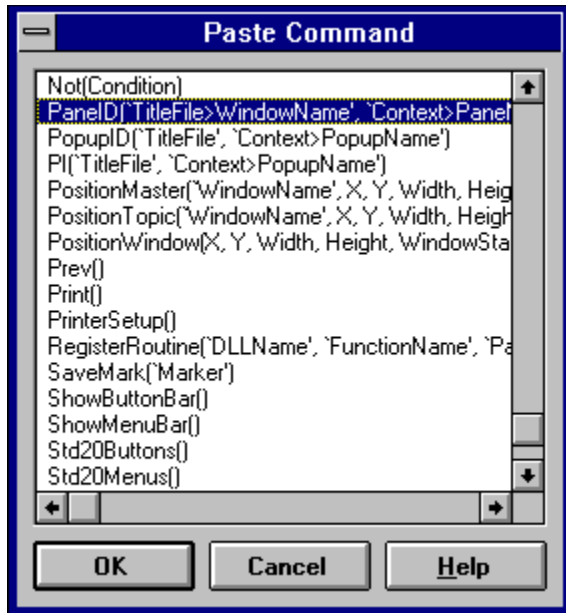
Dismiss When: Title is Closed

Coordinate System: Device-Independent (1024 x 1024) ...













**Edit Command**


**Edit Command:** PanelD

**TitleFile:**  

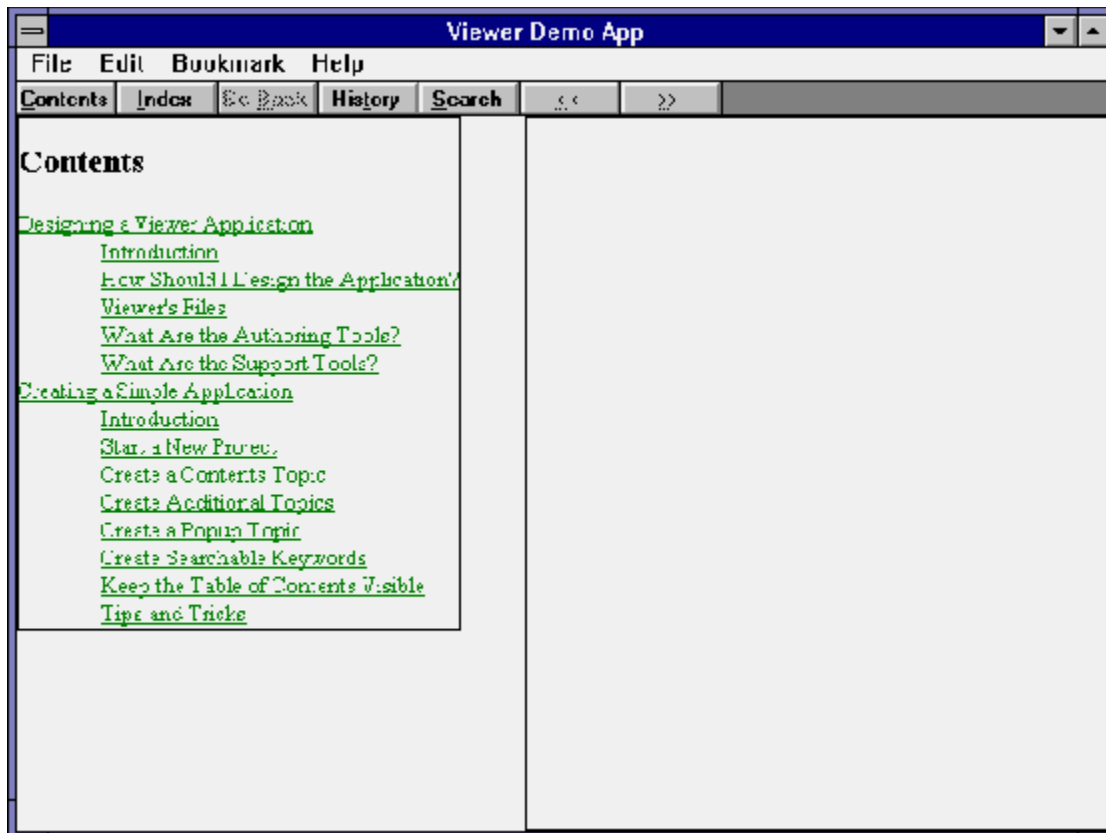
**WindowName:**  

**Context:**  

**PaneName:**  

**PrintTabCopyOrder:**  

PrintTabCopyOrder: Specifies the position of the pane for printing, copying, and tabbing





You hear the phrase “A picture is worth a thousand words” many times, for one simple reason—it’s true! A properly selected and placed picture makes a description much clearer and easier to understand. That’s why this book is filled with pictures. They make it easier for you to know exactly what you should see on your screen. If you see something different, you know immediately that you did something wrong. You can find and correct the error quickly at that point. The same idea applies to your application. You need the right mix of pictures, text, and multimedia to make your application as effective and enjoyable as possible.

You’re probably thinking “What’s the big deal? Why is there a whole chapter about this? After all, I have pasted pictures into documents by just using the Picture command from Word’s Insert menu. That certainly was easy. What else is there?”

There is a lot more. It’s easy to paste in a picture, but that doesn’t give you any way to do some extra things you might want to. Viewer has a powerful command that lets you

- ü Position the picture at either margin, with text wrapping around the picture
- ü Position the picture in the body of the text, aligned with the characters alongside
- ü Insert a caption, and control its position—above or below, centered, or aligned with the left or right edge of the picture; you can even control the color of the caption text and background
- ü Use a 256-color picture on systems that can display it, and control how it is shown on 16-color systems
- ü Make sure your picture looks its best on different displays—VGA, SVGA, or 8514
- ü Cause Viewer to execute one or more commands when the user clicks on the picture; you can even execute different commands if the user clicks on different parts of a single picture

This chapter shows you how to do all of these things and more. You also learn how to show a picture only when the user requests it, in popup or secondary windows. Finally, you learn about various problems you might run into, and how to prevent or cure them.

There is one other important reason to use the Viewer command instead of the crude technique of pasting a picture. Viewer can’t compile a topic that is larger than 32K bytes. Pictures that are pasted in a Word document count toward this size limitation, while pictures displayed with a Viewer command do not. A 320x200x256 color image contains 64,000 bytes, while a 160x100x16 color picture contains 8,000 bytes. Thus, one pasted picture with 256 colors, or a few with 16 colors, could be enough to bring you over the limit.

## Insert a Picture in a Topic?

Complexity: EASY

### Problem

I want to include pictures within my topics. Some of them should go in the middle of my text, others at the left margin, and others at the right margin. Text should flow around pictures that are positioned at the margins. The pictures can't add lots of bytes to the topic size, or my topic will be too big.

### Technique

You use Topic Editor to insert a picture and control its position. Text always wraps around pictures that are positioned at either margin.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP4\_1, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP4\_1. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the picture file for this How-To from the VIEWERHT\HOWTOS\CHAP4\CHAP4\_1\PICTURES directory on the CD-ROM to the VIEWERHT\CHAP4\_1\PICTURES subdirectory on your hard drive.
3. Use Project Editor to create a new project file in your VIEWERHT\CHAP4\_1 directory. Enter the name of your document file as TEXT\CHAP4\_1.RTF. Save the updated project file.
4. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this demonstration doesn't need context strings or other topic features because it only uses one topic. Viewer displays the first topic when a file is loaded unless you specify otherwise.

Now you can create the document file:

5. Enter the following text:

This text was before the command. This text is after the command. Just to show what happens, we include lots of text here. Did you know that Viewer applications can be created to run under Windows 3.1 or Modular Windows, and on Sony Multimedia CD-ROM Players and Tandy Video Information Systems? The Topic Editor command you use in this How-To can prepare pictures for any of these environments.

6. Place the insertion point between the first and second sentences of your text. Call up Topic Editor with the hotkey you selected (probably [CTRL]-[T]), and select the Picture (using ewX with {vflid137438953483}MVBMP2{vflid-3532125269655520}) entry. This brings up the standard Topic Editor window showing the command and



its elements.

7. Click on the Options button to bring up the Picture Options dialog box. This lets you select the picture file and control its position and appearance on other systems. (It does a lot more besides, as you see throughout this chapter.)
8. Click on the box with the ellipses (...) at the end of the Picture Filename field to bring up the Select Picture dialog box. This looks and works just like a standard File Open dialog box. Use this to select the picture file to be displayed as CHAP4\_1.BMP. Click on OK when you are done, to return to the Picture Options dialog box.

Next set the options to control how the picture is displayed:

9. Click on the Store Picture in Baggage check box. (If you are not familiar with the Baggage file system, it is explained in section 2.2.)
10. Leave the Display Error Message box checked. This tells Viewer to display a meaningful error message if the picture cannot be shown.
11. Click on the Position button that represents the desired justification for this picture: Left, Right, or Text Aligned. Choose Left in this example.
12. The completed dialog box should look like Figure 4-1. You learn the other options later. For now, click on OK.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½1 "[Macro=JI('viewerht.mvb>SecWin', `fig4_1')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

13. You see that the Topic Editor window now has the file name and options you selected. It shows the ewl command because you selected left-aligned. It would be ewr if you had clicked on right alignment or ewc for character alignment. The term ewX is used to refer to the group of commands.
14. Click on OK again, and you see the command appear in the Word document. It should look like Figure 4-2.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½2 "[Macro=JI('viewerht.mvb>SecWin', `fig4_2')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

15. By putting the picture in the middle of the text, you can see the text automatically wrapping around the picture. Of course, you could just type the Viewer ewX command right into your document, but it's a lot easier to use Topic Editor.
16. Now you can compile and display your application. It should look like Figure 4-3.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½3 "[Macro=JI('viewerht.mvb>SecWin', `fig4_3')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

The ewX command creates an embedded window and calls the Viewer function MVBMP2 to display the picture you selected. The ewl, ewr, or ewc command is used, based on the position option you select. The other options chosen are all passed to the function to use in filling the window. The exclamation point before the file name indicates that the file is stored in Baggage.

Topic Editor and Project Editor usually work together. When you selected Store Picture in Baggage you did more than set the option in the ewX command. You also set the option in the project file that causes the picture file to be copied into Baggage when your application is compiled.

### Comment

You noticed the lack of a margin between the picture and the wrapping text, right? It would look better if there was some space between them. With Viewer you currently only have three ways to do this—and none of them is all that great.

One way is to include a white border within your bitmap file, but this stands out if the user selects a different background color on his or her system. A variation is to set the Background color of the window and all the panes to match the color of the border in the picture. This looks better, but the user may be upset that your application doesn't use the colors that were set on his or her computer.

A second way is to put the ewX command and the text in adjacent cells in a table, but this doesn't let the text wrap around the top and bottom of the picture.

A third solution is to position the picture on the right border then add space by putting extra blanks or soft carriage returns [Shift]-[Enter] in the text at the necessary places. This looks better than any of the options available for a left-margin picture, but it's a nuisance. If you create the space in your text, it does not appear correctly if the user resizes the window and Viewer reformats your text to match the new size. In short, there aren't any ideal solutions.

If the text starts after the ewl command, it displays starting at the upper-right corner of the picture, without wrapping around the top of the picture. If the command is at the end of the paragraph then all text precedes the picture without wrapping. When the command is in the middle of the paragraph, as in this demonstration, the picture is displayed at the beginning of the text line following the location of the command.

Try creating variations on this example to see what happens. Use all three picture alignments, with text before and after each. Try combinations of large pictures with very little text and the reverse. Use your imagination! This type of experimenting is the best way to learn all the capabilities of Viewer.

## Display Multiple Pictures in a Row?

Complexity: INTERMEDIATE

### Problem

I want to create my own button bar, by placing a series of pictures close together in a row or column. I may want to use captions to label each button. I want to use these bars as my user interface.

### Technique

You can do this a few different ways in Viewer. First, you create a 1x6 (horizontal) table and put a picture in each cell. Then you do the same thing with and without tables and with and without captions. You use the Viewer ewc command with the MVBMP2 function, as you did in the last example.

You won't see all of the details of using these bars as a user interface—this is covered in detail in Chapter 6—but you get ready for that. At the end of this section you see the minor changes that are needed to create a vertical button bar.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP4\_2, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP4\_2. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the picture files for this How-To from the VIEWERHT\HOWTOS\CHAP4\CHAP4\_2\PICTURES directory on the CD-ROM to the VIEWERHT\CHAP4\_2\PICTURES subdirectory on your hard drive.
3. Use Project Editor to create a new project file in your VIEWERHT\CHAP4\_2 directory. Enter the name of your document file as TEXT\CHAP4\_2.RTF. Save the updated project file.
4. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this demonstration doesn't need context strings or other topic features because it only uses one topic. Viewer displays the first topic when a file is loaded unless you specify otherwise.

Now you can proceed with this How-To:

5. Position the insertion point in Word where you want the horizontal bar.
6. Click on Word's Table icon and draw a 1x6  
{vfld137438953482}table{vfld-137980119351296}; that is, a table with six cells across and one down, as shown in Figure 4-4.

```
{ewc vwrht2, TsTextButton, "Figure
4i½4"[Macro=JI('viewerht.mvb>SecWin', `fig4_4')] [Font="Arial"
```

/S12/B4] /W100 /H40/B1/D2}

7. Put the insertion point in the first cell of your table.
8. Call up Topic Editor with your hotkey, and select the Picture (using ewX with {vfld137438953483}MVBMP2{vfld-8970462913399619584}) command. This brings up the standard Topic Editor window showing the command and its elements.
9. Next, click on the Options button to bring up the Picture Options dialog box. This dialog box lets you select the picture file and control its position and appearance on other systems.
10. Click on the box with the ellipses (...) after the Picture Filename to display the Select Picture dialog box. This looks and works just like a standard File Open dialog box. Use this to select the file to be displayed as EDITCOPY.BMP. Click on OK when you are done, to return to the Picture Options dialog box.

Next set the options to control how the picture is displayed:

11. Click on the Store Picture in Baggage box. (If you are not familiar with the Baggage file system, it is explained in section 2.2.)
12. Leave the Display Error Message box checked. This tells Viewer to display a meaningful error message if the picture cannot be shown.
13. Leave the Text Aligned position selected.
14. Click on the Caption button to bring up the Picture Caption dialog box.
15. Enter the caption text Edit[ENTER]Copy.
16. Select the Centered Text Alignment and Caption Position Below options.
17. Leave the foreground and background colors alone for now. If you click on the boxes with the ellipses (...) you see the options. The completed dialog box should look like Figure 4–5.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½5"[Macro=JI('viewerht.mvb>SecWin', `fig4_5')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

18. Click on the OK button here and in each of the other dialog boxes until you are back in Word.
19. Repeat steps 8 through 15 for each cell in your table. Select a different picture file for each cell, and use an appropriate caption for each. When you are done your Word document should look like Figure 4–6.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½6"[Macro=JI('viewerht.mvb>SecWin', `fig4_6')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

20. Now compile the file. It should look similar to Figure 4–7.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½7"[Macro=JI('viewerht.mvb>SecWin', `fig4_7')] [Font="Arial"
```

/S12/B4] /W100 /H40/B1/D2}

### More Ways to Make a Button Bar

The button bar entries are awfully far apart, aren't they! You want it to look better than that. Let's try doing this in different ways—without the table, without captions, or without either.

21. Double-click on the text file name to start Word.
22. Insert a few blank lines after the table created in the previous steps.
23. Use Topic Editor, as in steps 8 through 20, to set up the same commands without using a table. Before starting each command, be sure the insertion point is immediately after the closing brace from the previous command. These entries should look like Figure 4-8.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½8"[Macro=JI('viewerht.mvb>SecWin', `fig4_8')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

24. Now compile your application, and use the Browse button to view the second topic. This version should look like Figure 4-9. It certainly looks a lot better!

```
{ewc vwrht2, TsTextButton, "Figure  
4i½9"[Macro=JI('viewerht.mvb>SecWin', `fig4_9')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

25. Let's see how it looks if you leave out the captions. Insert a few blank lines after the previous version, then create another table as in steps 5 through 19, but don't enter any captions. Figure 4-10 shows the result.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½10"[Macro=JI('viewerht.mvb>SecWin', `fig4_10')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

26. Repeat steps 22 and 23 without entering any picture captions. Figure 4-11 shows all of the versions, including this result. It finally looks a lot like a normal button bar, doesn't it?

```
{ewc vwrht2, TsTextButton, "Figure  
4i½11"[Macro=JI('viewerht.mvb>SecWin', `fig4_11')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

Just as in How-To 4.1, this method uses an ewX command to create an {vfld137438953482}embedded window {vfld8286622773195833344} and call the Viewer MVBMP2 function to display the selected picture. The ewl,

ewr, or ewc command is used, based on the position option you select. The other options chosen are all passed to the function to use in filling the window. The exclamation point before the file name indicates that the file is stored in Baggage. In this example you used of the caption option and tables to arrange multiple pictures.

You saw that using a table increases the space between pictures. Actually, that doesn't *have* to be true. Word defaults to setting the width and height of the table cells automatically, based on the contents. In this How-To, this caused Word to make the cells large enough to hold your commands. You can force the column width, row height, and spacing between columns as desired through Word's Table menu. This is a real nuisance, since you don't know in advance how big to make them—you have to use trial and error to get the appearance you want.

Why would you ever want to put pictures in a table, then? One important benefit is that each picture in the table takes up the same amount of space. That can be important if you occasionally want to replace some of the pictures while leaving others unchanged. You need to keep all of the pictures in the same locations so they won't appear to jump around between topics. You see this technique used when you build your own user interface in Chapter 6.

Topic Editor and Project Editor usually work together. When you selected Store Picture in Baggage you did more than set that option in the ewX command. You also set the option in the project file that causes the picture file to be copied into Baggage when your application is compiled.

### **Comment**

You are also interested in creating a vertical button bar. This requires creating a vertical table, such as 6x1. Since your only position options for the caption through Topic Editor are above or below the picture, you can't make this look as good as you'd like. You'd probably use a 6x2 table and put the captions in the adjacent cells through Word instead. If you didn't like the effect of the table, you could just place an ewc command on each line.

These buttons don't do anything if the user tries to click on them, but that's obviously needed. You get to that in sections 4.4 and 4.5.

Make sure all your pictures are the exact same height to avoid a jagged appearance. The widths should also be the same if possible. If not, make the difference in width large enough to be clearly deliberate.

**Pop Up a Picture?**

Complexity: EASY

**Problem**

I want to display a picture, but only when the user clicks on a hot spot in my document.

**Technique**

There are two different methods of doing this in Viewer. One displays the picture in a popup window, which only remains visible until the user clicks anywhere outside the picture. The second method displays the picture in a secondary window, which remains open until the user closes it.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

**Steps**

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP4\_3, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP4\_3. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the picture file for this How-To from the VIEWERHT\HOWTOS\CHAP4\CHAP4\_2\PICTURES directory on the CD-ROM to the VIEWERHT\CHAP4\_2\PICTURES subdirectory on your hard drive.
3. Use Project Editor to create a new project file in your VIEWERHT\CHAP4\_3 directory. Enter the name of your document file as TEXT\CHAP4\_3.RTF. Save the updated project file.
4. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this demonstration doesn't need context strings or other topic features because it only uses one regular topic. Viewer displays the first topic when a file is loaded unless you specify otherwise.

Now you can proceed with this How-To:

5. First define the popup hot spot. In the new topic, type in the hot spot text you want the user to see, such as **Pop up a picture**.
6. Select the hot spot text you just entered.
7. Invoke Topic Editor with your hotkey.
8. A list of appropriate functions is displayed. You want the first, Hot spot (text), which is already selected. Click on the OK button.
9. For Hot Spot Type, select Popup, as shown in Figure 4-12.

```
{ewc vwrht2, TsTextButton, "Figure
4i½12"[Macro=JI('viewerht.mvb>SecWin', `fig4_12')] [Font="Arial"
```

/S12/B4] /W100 /H40/B1/D2}

10. Then select the Jump to element to display the popup options.
11. Enter the context string you use for the popup topic: `ctx_pop_picture`.
12. Leave the Window and MVB Filename entries blank. The completed dialog box should look like Figure 4–13.

```
{ewc vwrht2, TsTextButton, "Figure  
4½13"[Macro=JI('viewerht.mvb>SecWin', `fig4_13')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

13. Click on OK in each dialog box until you are back in Word.

Next, prepare the picture to be displayed by creating a topic that only contains that picture. It could contain a caption and/or normal text as well, but leave them out to keep this simple.

14. Define a new topic after the current one, with context string `ctx_pop_picture`. No title or other options should be defined.
15. Define an `{vfld137438953482}` embedded window `{vfld-8970462913399619584}`, displaying the picture `CHAP4_3.BMP`, immediately after the # footnote. See How–To 4.1 if you haven't created an embedded window before.
16. Save the file and compile your application as usual.
17. The compiled result looks like Figure 4–14 after you click on the hot spot. Notice the shadow below and to the right of the picture, helping to set it off. Viewer decides where the popup appears, unless you define a custom popup window.

```
{ewc vwrht2, TsTextButton, "Figure  
4½14"[Macro=JI('viewerht.mvb>SecWin', `fig4_14')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Next try it the other way, using a `{vfld137438953482}` secondary window `{vfld71919613918576640}`. You can use the same popup topic you just created, so all you have to do is to define the secondary window and create a new hot spot.

18. To define the secondary window, use Project Editor, and choose Window Definitions from the Section menu.
19. Click on the New button to create a new window, then click on Properties to define the name and features of the new window.
20. Enter the Window Name as `SecWin`.
21. Enter How–To 4.3 Secondary Window as the Caption.
22. Set both the Height and Width fields to 425. The completed dialog box is shown in Figure 4–15.

```
{ewc vwrht2, TsTextButton, "Figure
```



4i½15"[Macro=JI('viewerht.mvb>SecWin', `fig4\_15')] [Font="Arial" /S12/B4] /W100 /H40/B1/D2}

23. Use the Windows Control box at the upper left corner to close the window, then click on OK in the Window Definitions dialog box.

Next create the hot spot:

24. Use Project Editor to start Word for your document, and position the insertion point just below the previous hot spot.
25. Type in the hot spot text you want the user to see, such as **Pop up a picture in a secondary window.**
26. Select the hot spot text you just entered.
27. Call up Topic Editor with your hotkey.
28. A list of appropriate functions is displayed. You want the first, Hot spot (text), which is already selected. Click on the OK button.
29. Leave the Hot Spot Type as Normal this time, and select the Jump to element to display the popup options.
30. Enter the same context string you used before, `ctx_pop_picture`.
31. Pull down the list of Window names and select `SecWin`. You defined the window before the hot spot so that the window name would be available here. That prevents spelling errors that could occur if you had to type in the name. Leave the MVB Filename entry blank. The completed dialog box should look like Figure 4–16.

{ewc vwrht2, TsTextButton, "Figure  
4i½16"[Macro=JI('viewerht.mvb>SecWin', `fig4\_16')] [Font="Arial" /S12/B4] /W100 /H40/B1/D2}

32. Press OK in each dialog box until you are back in Word.
33. Save the file and compile your application as usual.
34. You should see something like Figure 4–17 when you test the new hot spot.

{ewc vwrht2, TsTextButton, "Figure  
4i½17"[Macro=JI('viewerht.mvb>SecWin', `fig4\_17')] [Font="Arial" /S12/B4] /W100 /H40/B1/D2}

### How It Works

The hot spot for the picture popup looks exactly like the one you created in Chapter 3. The displayed text is formatted in the document with a single underline (and displayed with a dotted underline in the compiled result), followed by the context string of the popup topic formatted as hidden.

The only thing that's different from the text popup you created before is that this time the popup topic contains a picture. It's that simple! Viewer sets the size and positions the popup window automatically.

The hot spot for the secondary window looks very much like a standard

jump hot spot. The displayed text is formatted in the document with a double underline (and displays with a single underline in the compiled result), followed by the topic context string formatted as hidden. The use of a secondary window is shown by following the context string with a {vfld137438953482}greater-than symbol {vfld-35184913254711296} (>) and the name of the secondary window. Figure 4-18 shows both of these hot spot definitions.

```
{ewc vwrht2, TsTextButton, "Figure  
4½18"[Macro=JI('viewerht.mvb>SecWin', `fig4_18')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

The choices you made while defining the secondary window appear in the Viewer MVP project file like this:

```
[WINDOWS]  
SecWin="How-To 4.3 Secondary Window", (100,100,300,300,0),  
(0),,,,,,(0),(0),(0)
```

This specifies the window's name, caption, position, size, and colors. The Viewer manuals have the complete documentation for this entry. There should never be any need to work with this directly, however—it's much easier to use Project Editor.

#### **Comment**

As you can see, it is equally simple to have a picture appear in a popup or secondary window. When you are deciding which to use, you need to understand the differences in what they do. Some major differences are:

- ü Popup windows are closed when the user clicks anywhere outside them. Secondary windows remain open until explicitly closed by the user (or when Viewer is closed).
- ü Viewer usually determines the size and position of popup windows, although you can control these features. (The use of an author-defined custom popup pane is demonstrated in How-To 5.4.) You set the size and position for secondary windows, and the user can move and resize them.
- ü None of the Viewer commands such as printing, copying to the clipboard, or jumping between topics can be performed within {vfld137438953484}popup windows {vfld13331578486784}. (You can have jump and popup hot spots within a popup, however.) Some commands, but not all, can be performed within {vfld137438953484}secondary windows {vfld280933810831360}. The distinction cannot be readily explained, but this information is included in Viewer's documentation.

Popups are usually used to help explain a single term or concept, where that explanation is only needed briefly. Secondary windows are usually used to display information that must remain visible as the user views different parts of the application.

Pictures can also be displayed in standard panes within the Viewer window. This technique is demonstrated in How-To 5.3.



## Use a Picture as a Hot Spot?

Complexity: INTERMEDIATE

### Problem

I want to use a picture, instead of text, as a hot spot. I want to be able to perform any function when the user clicks on the picture that I could with a text hot spot. I'd like to have all the other options for displaying pictures, as demonstrated throughout this chapter, available as well.

### Technique

You demonstrate two techniques. The first uses a normal hot spot with the Viewer `bmX` command in place of the displayed text. The hidden portion of the hot spot is no different than that of a standard text hot spot. This technique can be used to define jumps, popups, or secondary windows in the same way you have defined them in previous examples. This does not provide the display controls offered by the `ewX` command.

The second technique uses the `ewX` command that you having been learning throughout this chapter, and takes advantage of the option built into that command that operates just like a hot spot.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, `VIEWERHT\CHAP4_4`, and the standard subdirectories—`TEXT`, `SOUNDS`, `PICTURES`, `MOVIES`—under `CHAP4_4`. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the picture files for this How-To from the `VIEWERHT\HOWTOS\CHAP4\CHAP4_4\PICTURES` directory on the CD-ROM to the `VIEWERHT\CHAP4_4\PICTURES` subdirectory on your hard drive.
3. Use Project Editor to create a new project file in your `VIEWERHT\CHAP4_4` directory. Enter the name of your document file as `TEXT\CHAP4_4.RTF`. Save the updated project file.
4. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this demonstration doesn't need context strings or other topic features because it only uses one regular topic. Viewer displays the first topic when a file is loaded unless you specify otherwise.

Now you can proceed with this How-To. First use the `bmX` command:

5. Position the insertion point where you want the hot spot picture located in the topic.
6. Call up Topic Editor with your hotkey.
7. In the New Viewer Element dialog box, select  
`{vfld137438953482}Hot spot (picture){vfld-9042520507437547520}`

and click on OK to display Topic Editor's dialog box.

8. Select Normal as the Hot Spot Type. The dialog box should look like Figure 4-19.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½19"[Macro=JI('viewerht.mvb>SecWin', `fig4_19')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

9. Click on the Options button to display the Picture Options dialog box.
10. Click on the box with the ellipsis (...) after the Picture Filename to display the Select Picture dialog box.
11. Select the CHAP4\_4A.BMP file and click on OK to return to the Picture Options dialog box. The dialog box should look like Figure 4-20.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½20"[Macro=JI('viewerht.mvb>SecWin', `fig4_20')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

12. Click on OK to return to Topic Editor's dialog box.
13. Select the Jump to element to display the options dialog box.
14. Enter the context string of the topic you want to jump to as ctx\_jump.
15. Leave the Window and MVB Filename entries blank.
16. Leave the Hot Spot Formatting set to Underlined and Colored. The completed dialog box should look like Figure 4-21.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½21"[Macro=JI('viewerht.mvb>SecWin', `fig4_21')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

17. Click on OK to return to Word.

Next create a hot spot using the ewX command. You can compile them together.

18. Position the insertion point where you want this hot spot picture located, a few lines below the first one.
19. Call up Topic Editor with your hotkey.
20. In the New Viewer Element dialog box, select Picture (using ewX with {vflid137438953482}MVBMP2{vflid-8970462913399619584}), and click on OK.
21. In the Topic Editor dialog box, click on the Options button to display the Picture Options dialog box.
22. Click on the box with the ellipsis (...) after the Picture Filename to display the Select Picture dialog box.
23. Select the CHAP4\_4B.BMP file and click on OK to return to the Picture Options dialog box.

- Click on the Paste Command button to display the Paste Command dialog box. This dialog box lists all the commands that you can execute in a hot spot. Select the JumpID command as shown in Figure 4–22.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½22"[Macro=JI('viewerht.mvb>SecWin', `fig4_22')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

- Click on OK to return to the Picture Options dialog box.
- Next, click on the Edit Command button. This displays the Edit Command dialog box, where you can set the options used by the command you just pasted.
- Enter `CHAP4\_4.MVB' as the TitleFile surrounded by left and right single quotes ('). The left quote is found, on most keyboards, on the key with the tilde (~) to the left of the digit one (1). The right quote is usually found on the same key as the double quote, next to the Enter key.
- Enter main as the Window Name.
- Enter the context string `ctx\_jump' surrounded by left and right single quotes. The completed dialog box should look like Figure 4–23.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½23"[Macro=JI('viewerht.mvb>SecWin', `fig4_23')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

- Click on OK to return to the Picture Options dialog box. It should look like Figure 4–24 after you have made all of your selections.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½24"[Macro=JI('viewerht.mvb>SecWin', `fig4_24')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

- Click on OK to return to Topic Editor.
- Click on OK to return to Word. The entries in your Word document should look like Figure 4–25.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½25"[Macro=JI('viewerht.mvb>SecWin', `fig4_25')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

- Create a new topic with the context string ctx\_jump, and enter a few lines of text.
- Save your document as an RTF file and exit Word.
- Now compile and test your application. Both hot spots should look and work exactly the same.

## How It Works

You may have noticed that the `bmX` hot spot looks just like the familiar text hot spots. The only difference is that the `bmc` command is present where the displayed text is usually found.

The `ewc` command looks much like the ones you've seen in earlier examples in this chapter. You added the `[macro...]` section to define the action to be taken. If you look at it closely, you see that the entire `JumpID` macro is enclosed in `{vfld137438953482}` double quotes `{vfld11132555231232}`, and the variables within it are enclosed in `{vfld137438953482}` single quotes `{vfld-9223356093936173056}`. This is to distinguish the two logical levels of parameters. The single quotes *must* use a matched pair of left and right quotes.

### **Comment**

Since both commands look and work the same, what difference does it make which you use? The `ewX` command gives you some important added options. It lets you specify a caption, as you saw in How-To 4.3. It also lets you use 256-color pictures, as you see in How-To 4.6. The Paste Command and Edit Command dialog boxes also make it easy to invoke any other operations supported by Viewer besides jumps and popups. You learn some of these operations in later chapters.

You may have also noticed that the `{vfld137438953483}bmX{vfld-9042102693018992640}` command did not offer an option of storing the picture in the Baggage section. Files used by `bmX` commands are included in the `MVB` file by the compiler. If you use these commands, be sure your directory structure during authoring matches the structure you install on your users' machines. Notice in Figure 4-26 that the subdirectory name used during authoring is included in the command. This is part of the compiled application.

The `bmX` command is supported primarily for compatibility with Viewer 1.0 and Windows Help applications, so that this command does not *have* to be converted.

There is one critical design consideration that applies to any graphic hot spot—how does the user *know* it's a hot spot? Text hot spots are normally identifiable by their green color and underline, but you can't do the same thing with a picture. The cursor changes to a pointing finger over the entire picture, as it does over text hot spots, but it's risky to count on your users noticing that. A better answer would be some indication in a caption. You have to choose your own solution. Remember that it should be consistent throughout your application, and either be self-evident (that's tough to do!) or be explained at the start of the program.

## Use Multiple Hot Spots in One Picture?

Complexity: INTERMEDIATE

### Problem

I want to include one big picture, and have different things happen when the user clicks on different parts of it.

### Technique

You use Viewer's Segmented Hotspot Editor (`{vfld137438953482;SHED2{vfld3940108508069888}`) to define multiple hot spots within a single picture. The resulting file is then incorporated into the Viewer application through the `ewX` command. You use a screen capture of the Program Manager group for Viewer. You define each icon in the group as a separate hot spot.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, `VIEWERHT\CHAP4_5`, and the standard subdirectories—`TEXT`, `SOUNDS`, `PICTURES`, `MOVIES`—under `CHAP4_5`. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the picture file for this How-To from the `VIEWERHT\HOWTOS\CHAP4\CHAP4_5\PICTURES` directory on the CD-ROM to the `VIEWERHT\CHAP4_5\PICTURES` subdirectory on your hard drive.
3. Use Project Editor to create a new project file in your `VIEWERHT\CHAP4_5` directory. Enter the name of your document file as `TEXT\CHAP4_5.RTF`. Save the updated project file.

Now you can proceed with this How-To:

4. First you must start the `SHED2` program and load the `CHAP4_5.BMP` picture file. Maximize the window containing your picture for easiest visibility.
5. Mark each of your hot spot regions by clicking on one corner and dragging to the opposite corner. The outline is displayed as you drag, as shown in Figure 4-26. The Edit menu shows an item named Undo New Hot spot after a new hot spot region is defined. You can use that if you make a mistake.

```
{ewc vwrht2, TsTextButton, "Figure
4i½26"[Macro=JI('viewerht.mvb>SecWin', `fig4_26')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

6. Select the first region by clicking within it.
7. Choose Attributes from the Edit menu to display the corresponding



dialog box.

8. Set the Hot Spot Type to Popup for this example.
9. Enter the context string `ctx_viewer`. The completed dialog box should look like Figure 4–27.

```
{ewc vwrht2, TsTextButton, "Figure  
4i½27"[Macro=JI('viewerht.mvb>SecWin', 'fig4_27')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

10. Click on OK when done.
11. Repeat these steps for each hot spot region defined. The context strings are all abbreviations of the application names, as follows.  
`ctx_compiler`, `ctx_project_editor`, `ctx_hotspot_editor`, `ctx_wave_edit`,  
`ctx_bit_edit`, `ctx_pal_edit`, `ctx_convert`, `ctx_viewer_api`,  
`ctx_compiler_help`, `ctx_authoring_help`, `ctx_read_me`,  
`ctx_common_questions`, `ctx_authoring_guide`, `ctx_gallery`
12. When all of the regions have been defined, save the file as `CHAP4_5.SHG` and exit.

Next create the document file:

13. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this demonstration doesn't need context strings or other topic features because it only uses one regular topic. Viewer displays the first topic when a file is loaded unless you specify otherwise.
14. Place the insertion point where you want the picture displayed.
15. Call up Topic Editor with your hotkey, and select the Picture (using `ewX` with `MVBMP2`) command. This brings up the standard Topic Editor window showing the command and its elements.
16. Next, click on the Options button to bring up the Picture Options dialog box. This dialog box lets you select the picture file and control its position and appearance on other systems.
17. Click on the box with the ellipsis (...) after the Picture Filename to display the Select Picture dialog box. This looks and works just like a standard File Open dialog box. Use this to select file `CHAP4_5.SHG`. Click on OK when you are done to return to the Picture Options dialog box.
18. Click on the Store Picture in Baggage box.
19. Leave the Display Error Message ... box checked.
20. Leave the position as Text Aligned.
21. Click on the Caption button to bring up the Picture Caption dialog box.
22. Enter This picture has multiple hot spots as the caption text, and choose Centered text alignment and caption position Below. Press OK when done to return to the Picture Options dialog box.
23. Click on OK in each dialog box until you are back in Word.
24. Create new topics for each of the 15 hot spots. Each must have one of the

context strings you used in defining the hot spots, and a single line of text describing the corresponding application. For example, the text for context string `ctx_viewer` might say This is the Viewer runtime program.

25. Save your document as usual and exit Word.
26. Compile and test your application. The picture does not show the borders of the hot spot regions, but the cursor changes to a pointing finger over each one. As you click on each icon, a popup window with your description of the application appears. It should look like Figure 4–28.

```
{ewc vwrht2, TsTextButton, "Figure  
4½"[Macro=JI('viewerht.mvb>SecWin', 'fig4_28')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

The SHED2 program stores the attributes for each region you define within the SHG file. This information is used by Viewer when the picture is displayed. The SHG file is displayed just like any other picture, except that it *cannot* be pasted. That doesn't matter, since you know that's not the best way to handle pictures.

### Comment

You can use 256-color pictures in SHED2. The next section demonstrates how to use 256-color pictures in Viewer.

You can display the Attributes dialog box in SHED2 by double-clicking within the desired hot spot area.

|   |
|---|
| Warning: The compiler does not report any nonexistent context strings used in the {vfld137438953484}SHG {vfld-9042102693018992640} file as it does for normal hot spots. Be sure to test all hot spot definitions after the file is compiled. |
|---|

## Use 256–Color Pictures?

Complexity: INTERMEDIATE

### Problem

I want to include some beautiful 256–color pictures, but not all of my users can display them. I need to have the pictures automatically adjusted to the capabilities of the user’s display so that they always look their best.

### Technique

You use two more features of the Viewer `ewX` command here. First is the option of specifying dither or MCGA processing for 256–color pictures being displayed on a 16–color system. The second feature is the ability to display entirely different bitmap files on 16–color or 256–color systems.

You create the standard directories, project file, and document file for this How–To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, `VIEWERHT\CHAP4_6`, and the standard subdirectories—`TEXT`, `SOUNDS`, `PICTURES`, `MOVIES`—under `CHAP4_6`. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the picture files for this How–To from the `VIEWERHT\HOWTOS\CHAP4\CHAP4_6\PICTURES` directory on the CD–ROM to the `VIEWERHT\CHAP4_6\PICTURES` subdirectory on your hard drive.
3. Use Project Editor to create a new project file in your `VIEWERHT\CHAP4_6` directory. Enter the name of your document file as `TEXT\CHAP4_6.RTF`. Save the updated project file.
4. Use Project Editor to start Word and create your document. You don’t need to define a topic—the beginning of the file is always a new topic, and this demonstration doesn’t need context strings or other topic features because it only uses one regular topic. Viewer displays the first topic when a file is loaded unless you specify otherwise.

Now you can proceed with this How–To:

5. Position the insertion point where you want the first version of your picture displayed.
6. Call up Topic Editor with your hotkey, and select the Picture (Using `ewX` with `MVBMP2`) command. This brings up the standard Topic Editor dialog box showing the command and its elements.
7. Click on the Options button to bring up the Picture Options dialog box. This lets you select the picture file and control its position and appearance on other systems.
8. Click on the box with the ellipsis (...) after the Picture Filename to display the Select Picture dialog box. This looks and works just like a

standard File Open dialog box. Use this to select the PICT256.BMP file. Click on OK when you are done, to return to the Picture Options dialog box.

Next set the options:

9. Click on the Store Picture in Baggage box.
10. Click on the Dither Picture... checkbox.
11. Leave the Display Error Message ... box checked.
12. Select the desired position as left aligned.
13. Click on the Caption button.
14. Enter caption text Dither Example and select the Center checkbox under Text Alignment. Click on OK to return. The completed dialog box should look like Figure 4–29.

```
{ewc vwrht2, TsTextButton, "Figure  
4½29"[Macro=JI('viewerht.mvb>SecWin', 'fig4_29')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

15. Click on OK in each dialog box until you return to Word.

Next set up the same picture, but use the MCGA option:

16. Position the insertion point a few lines below the previous command, and repeat steps 6 through 15, *except*:
17. Click on Allow User to View Bitmap in MCGA Mode instead of Dither Picture ....
18. Enter caption text as MCGA Example.

Next set up to display either of two picture files, based on the display's color capacity. The process is very similar, as it uses the same ewX command.

19. Position the insertion point where you want this picture displayed.
20. Call up Topic Editor with your hotkey, and select the Picture (using ewX with MVBMP2) command.
21. Click on the Options button to bring up the Picture Options dialog box.
22. Click on the 8-bit Display file folder file folder tab. You define the file and options for the 256-color picture here.
23. Click on the box with the ellipsis (...) after the Picture Filename to display the Select Picture dialog box. Use this to select the PICT256.BMP file. Click on OK when you are done, to return to the Picture Options dialog box.
24. Click on the Store Picture in Baggage check box.
25. Click on the Caption button.
26. Enter Caption text Different Pictures Example (256).
27. Click on Center position. The resulting dialog box should look like Figure 4–30.

```
{ewc vwrht2, TsTextButton, "Figure
```

4i½30"[Macro=JI('viewerht.mvb>SecWin', 'fig4\_30')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}

28. Click on OK to return.
29. Next click on the 4-bit Display folder file folder tab.
30. Select the 16-color file to be displayed as PICT16.BMP.
31. Click on the Store Picture in Baggage check box.
32. Click on the Caption button.
33. Enter Caption text Different Pictures Example (16).
34. Click on Center position.
35. Click on OK to return.
36. Now click on the Any Display folder, and set the options that apply to all systems.
37. Leave the Display Error Message ... box checked.
38. Select the desired position—left aligned.
39. Don't select a file name or enter a caption here.
40. Click on OK in each dialog box until you return to Word.
41. The commands in your document should look very much like the following:

Dither Example:                    { ewc MVBMP2, ViewerBmp2,  
   [caption="\pc;bDither Example"  
   dither]!pict256.bmp}

MCGA Example:                    { ewc MVBMP2, ViewerBmp2,  
   [mcca caption="\pc;bMCGA  
   Example"!pict256.bmp}

Separate BMPs  
Example:                            { ewc MVBMP2, ViewerBmp2,  
   [256color caption="\pc;bSeparate  
   BMPs Example (256)"!pict256.bmp  
   [16color caption="\pc;bSeparate  
   BMPs Example (16)"!pict16.bmp}

42. Now compile and test these commands. To test them you need to set your system to display 16 colors. The first and third examples should look extremely similar. The caption on the last picture indicates the mode you are in—16 or 256 colors—because you use different captions for each mode. The MCGA example should have an icon of a magnifying glass.
43. Click on the MCGA icon to see the picture in MCGA mode.
44. If you can display this demonstration in 16-color mode, you should see fairly accurate colors in the Dither example, and distorted colors in the other examples.

### How It Works

The MVBMP2 function provides all of its capabilities through parameters passed in the ewX command. The particular options used here are as follows.

- ü Dither specifies that 256 colors should be simulated in 16 colors by combining two or more colors to approximate the appearance of a missing color. The details of this process can be controlled by clicking on the Dither button in the Picture Options dialog box. The default values usually are acceptable.
- ü MCGA specifies that a simplified 16-color version of the picture should be displayed, with a magnifying glass icon. Clicking on the icon causes the picture to be displayed in MCGA mode (320x200 resolution with 16 colors).
- ü 256color specifies that the file, caption, and other options are displayed on to systems capable of displaying 256 colors.
- ü 16color specifies that the file, caption, and other options are displayed on to systems capable of displaying 16 colors.

### Comment

What do you think happens with the last example on a system that displays more than 256 colors? Because you didn't specify an action for such systems, no picture is displayed—even though it is more than capable of displaying your 256-color bitmap. Fortunately, this problem can be easily solved. First specify the cases you want to give special handling—in this case that's the 16-color definition. After the special cases, specify the Any Display setup with the options you want for all other cases. The latter applies by default to any system that does not fit a specified capacity. You usually specify the 16-color handling plus a default to cover any greater capacity.

The file specified in these commands can be a SHG, DIB, BMP, or WMF file.

You can't use MCGA mode for 256-color SHG files or popups. The MCGA button steals mouse clicks, which prevents using them for other purposes.

Some video systems and drivers report to Viewer that they are capable of handling 256 colors, even though 16-color operation has been selected. These systems are treated as 256-color systems by Viewer.

Viewer's BitEdit utility can convert 256-color pictures to 16 dithered colors. This makes it possible to create picture files that take less space, use simple display options, and retain nearly all of the quality of the 256-color original.

## 4.7 Tips and Tricks

### File Formats

- ⇒ The MVBMP2 function supports DIB, BMP, SHG, and WMF file formats. Note that there are versions of the DIB and WMF formats that are not supported, such as RIFF. In addition, some programs create BMPs in nonstandard formats. The nonstandard BMPs usually are not recognized until Viewer refuses to display a file. They generally can be converted by loading them into Windows Paintbrush and saving them from that program. The Convert utility that is part of the Viewer package handles most other conversions you require. It can convert RIFFs to supported forms, for example.
- ⇒ Convert and BitEdit can create DIB RLEs. These are smaller files than the SHG format, and also display faster and use less memory.

### File Handling

- ⇒ Pictures that are pasted into the document are compressed in the way that is specified for the entire document through Project Editor. If the same picture is pasted in more than one place, each occurrence is included in the compiled result. This greatly increases the size of the compiled file but allows faster access to display the picture.
- ⇒ Pictures that are included in Baggage are not compressed, regardless of any {vfld137438953484}compression{vfld7882987656592752640} specified for the application in Project Editor. The smallest supported file formats, such as WMF, should be used whenever possible.
- ⇒ The best, if not only, program for converting bitmaps to metafiles is reported to be Corel Draw. However, it is too expensive for most authors to purchase for just the one function.
- ⇒ Always place your files in the Baggage area, as you did in this chapter, unless you have a very good reason for keeping them as separate files. Having one large file makes your application easier to install. It also helps prevent problems that would occur if one of your files is accidentally erased or overwritten.

### Picture & Monitor Resolutions

- ⇒ Many Windows programs, including the Help system, stretch bitmaps that were created in different resolutions to maintain the original logical size. This attempt to be helpful generally creates strange-looking pictures. The ewX command does not change the size of the picture. It uses the original bitmap without regard to the logical size. This means that a given bitmap occupies less of the screen on a 1024x768 (8514 mode) display than on a VGA. This is in a consistent proportion to the rest of the Viewer window in all cases. Note that bmX commands *do* cause bitmaps to be stretched.
- ⇒ Many modern video boards support a wide variety of resolutions, from 640x480 to 1280x1024, and allow the user to change between them fairly easily. If possible you should run your application in a wide variety of resolutions (and color capacities) to make sure it always looks the way

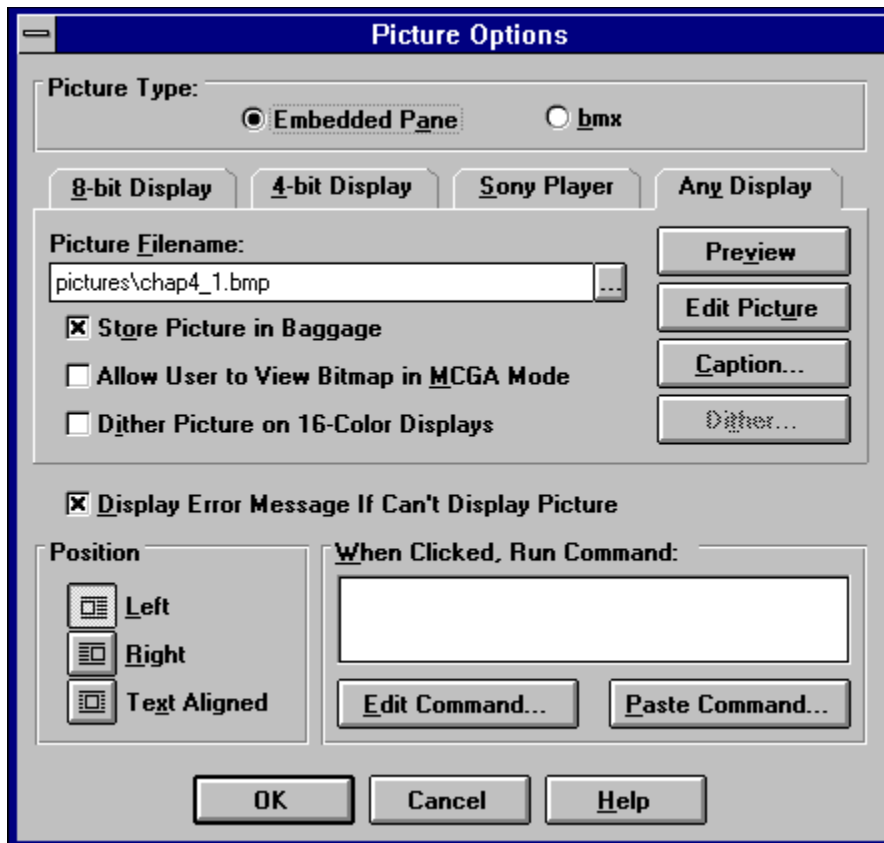
you want. This lets you catch problems before your users see them.

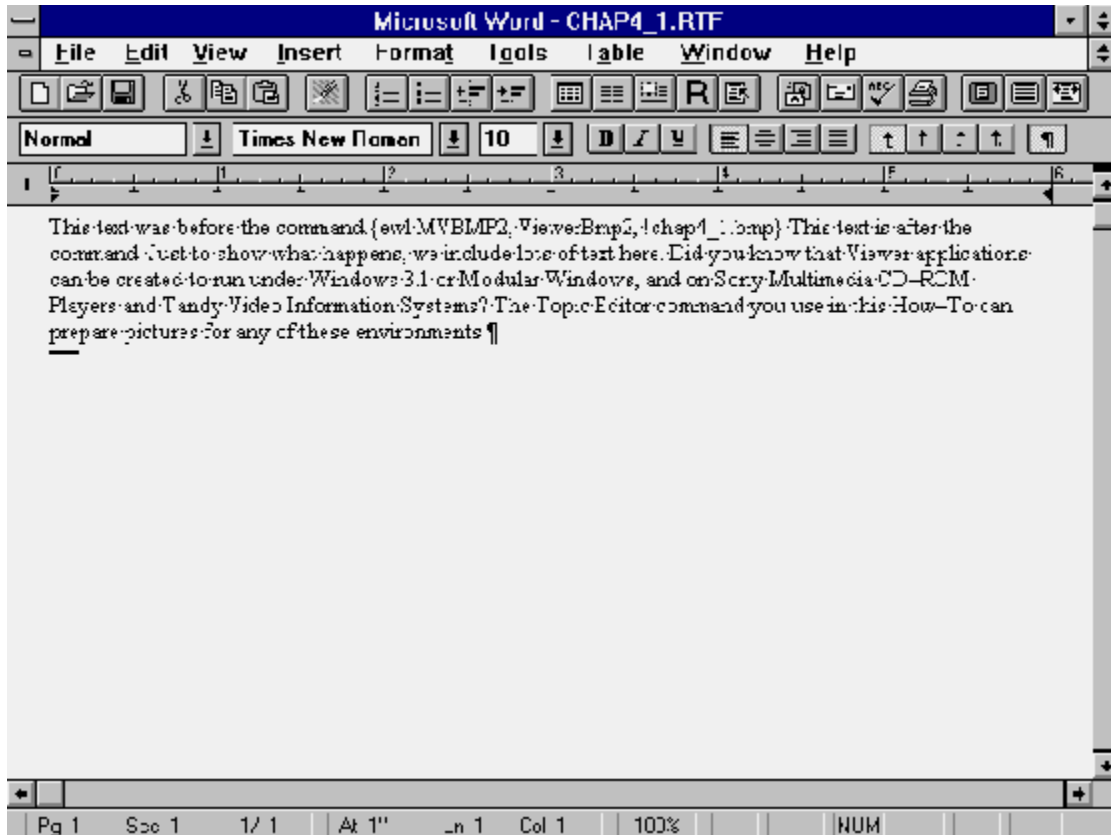
- ⇒ Scanners can capture images at far higher resolutions than the monitors that display the pictures. Although Viewer usually displays these images accurately, such images should be tested in a variety of display resolutions.
- ⇒ Viewer 1.0 and the Windows Help system used the Multiple Resolution Bitmap Compiler (MRBC) to get around the problems of picture resolutions that were different from screen resolutions. It allowed you to take versions of a bitmap that were created in different resolutions and combine them into a single file. The proper version of the image would be selected and displayed when the application was executed. This utility and its file format are not supported by Viewer 2.0 because they are no longer necessary.

### **Diagnosing Problems**

- ⇒ If an error message displays instead of the expected picture, the message text should give you a clue as to the reason. If it says the file could not be found, look at your Baggage section through Project Editor. If there is a blank line before the end, point to that entry and use the Project Editor menu to delete the entry. Blank lines can be caused by partially creating a command and then quitting. They have been known to cause all following lines in the section to be ignored.
- ⇒ Any time you have problems you can't explain, try exiting Project Editor and examining the MVP file in Notepad. If you see anything that looks wrong, save a copy of the original file under a different name and edit the project file through Notepad. Then re-execute Project Editor and see if your application compiles and runs correctly.
- ⇒ The compiler only reports nonexistent context strings that are used in standard hot spots. Context strings referenced in SHG files, ewX commands, macros, and other places are not checked for validity. Therefore, keep a list of all context strings, showing where they are defined and where they are used. All hot spots and other methods of executing jumps, popups, or macros should be tested as soon as possible. They should all be retested before shipping your final product in case any referenced topics were deleted.
- ⇒ Some authors have reported problems using Paintbrush with 256-color pictures. It loses the palette or reverses black and white colors. If you run into this you might have to use Viewer's BitEdit utility or some other drawing program.





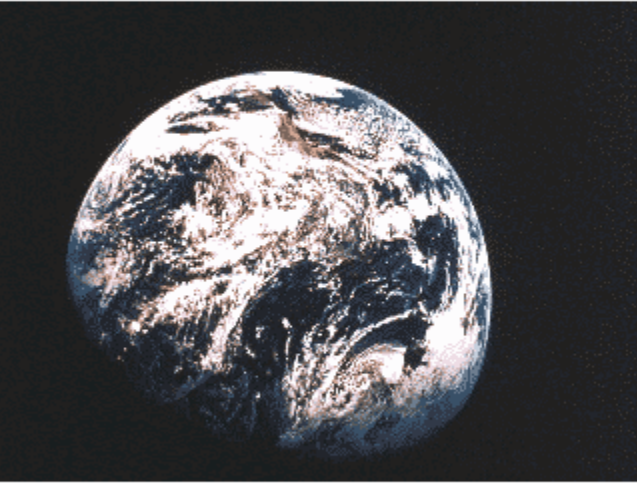


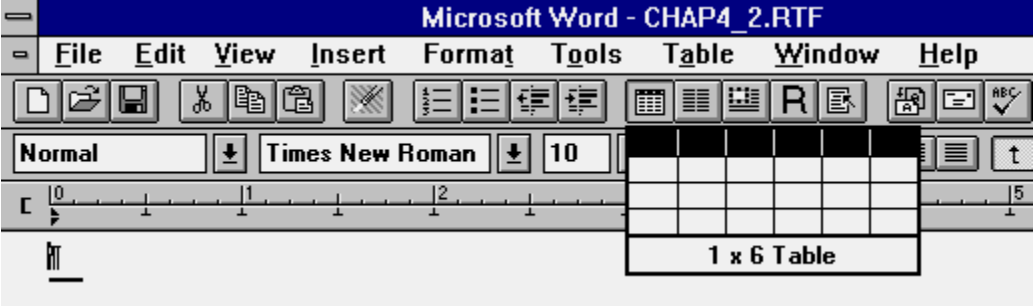
How-To 4.1 Demo

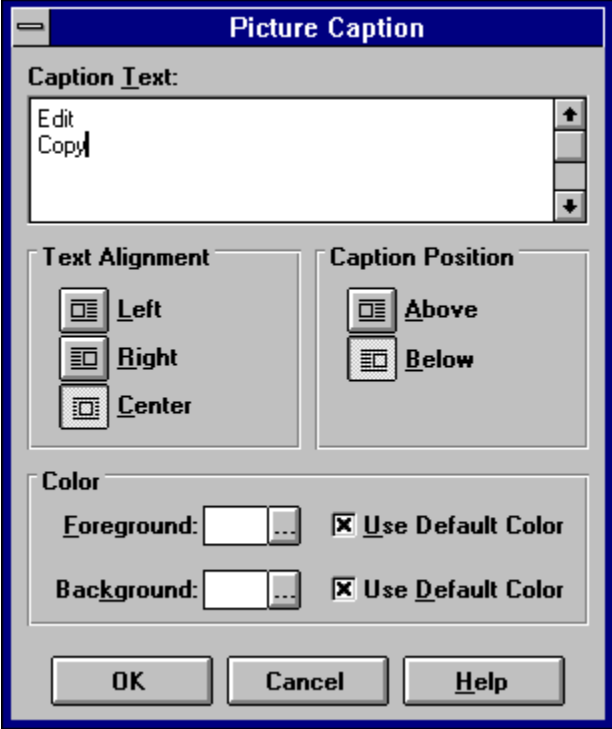
File Edit Bookmark Help

Contents Go Back History << >>

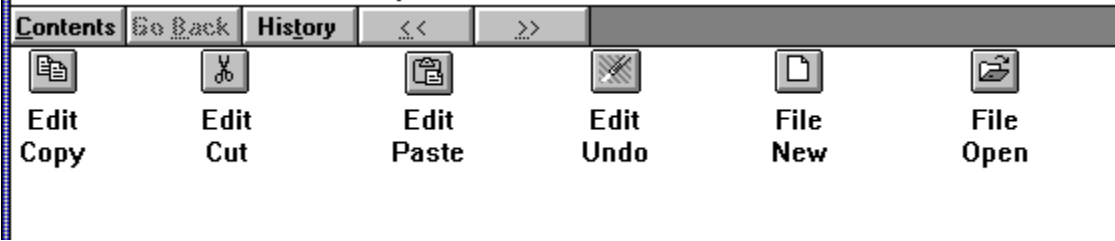
This text was before the command. This text is after the command. Just to show what happens, we include lots of text here. Did you know that Viewer applications can be created to run under Windows 3.1 or Modular Windows, and on Sony Multimedia CDROM Players and Tandy Video Information Systems? The Topic Editor command you use in this HowTo can prepare pictures for any of these environments.







| 0  | 1  | 2   | 3  | 4   | 5  |
|--|--|---|--|---|--|
| {ewc:<br>MVBMP2.<br>ViewerBmp2,<br>[caption="\pc;<br>bEdit\Copy"]!<br>editcopy.bmp}⊗ | {ewc:<br>MVBMP2.<br>ViewerBmp2,<br>[caption="\pc;<br>bEdit\Cut"]!<br>editcut.tmp}⊗ | {ewc:<br>MVBMP2.<br>ViewerBmp2,<br>[caption="\pc;<br>bEdit\Paste"]!<br>editpast.bmp}⊗ | {ewc:<br>MVBMP2.<br>ViewerBmp2,<br>[caption="\pc;<br>bEdit\Undo"]!<br>editundo.bmp}⊗ | {ewc:<br>MVBMP2.<br>ViewerBmp2,<br>[caption="\pc;<br>bFile\New"]!<br>ilenev.bmp}⊗ | {ewc:<br>MVBMP2.<br>ViewerBmp2,<br>[caption="\pc;<br>bFile\Open"]!<br>fileopen.bmp}⊗ |



Contents

Go Back

History

<<

>>



Edit  
Copy



Edit  
Cut



Edit  
Paste



Edit  
Undo



File  
New



File  
Open

```
¶
{ ewc: MVBMP2, ViewerBmp2, [caption="\pc;bEdit\Copy"!editcopy.bmp} { ewc: MVBMP2,
ViewerBmp2, [caption="\pc;bEdit\Cut"!editcut.bmp} { ewc: MVBMP2, ViewerBmp2,
[caption="\pc;bEdit\Paste"!editpast.bmp} { ewc: MVBMP2, ViewerBmp2,
[caption="\pc;bEdit\Undo"!editundo.bmp} { ewc: MVBMP2, ViewerBmp2,
[caption="\pc;bFile\New"!filenew.bmp} { ewc: MVBMP2, ViewerBmp2,
[caption="\pc;bFile\Open"!fileopen.bmp} ¶
¶
```





Edit  
Copy



Edit  
Cut



Edit  
Paste



Edit  
Undo

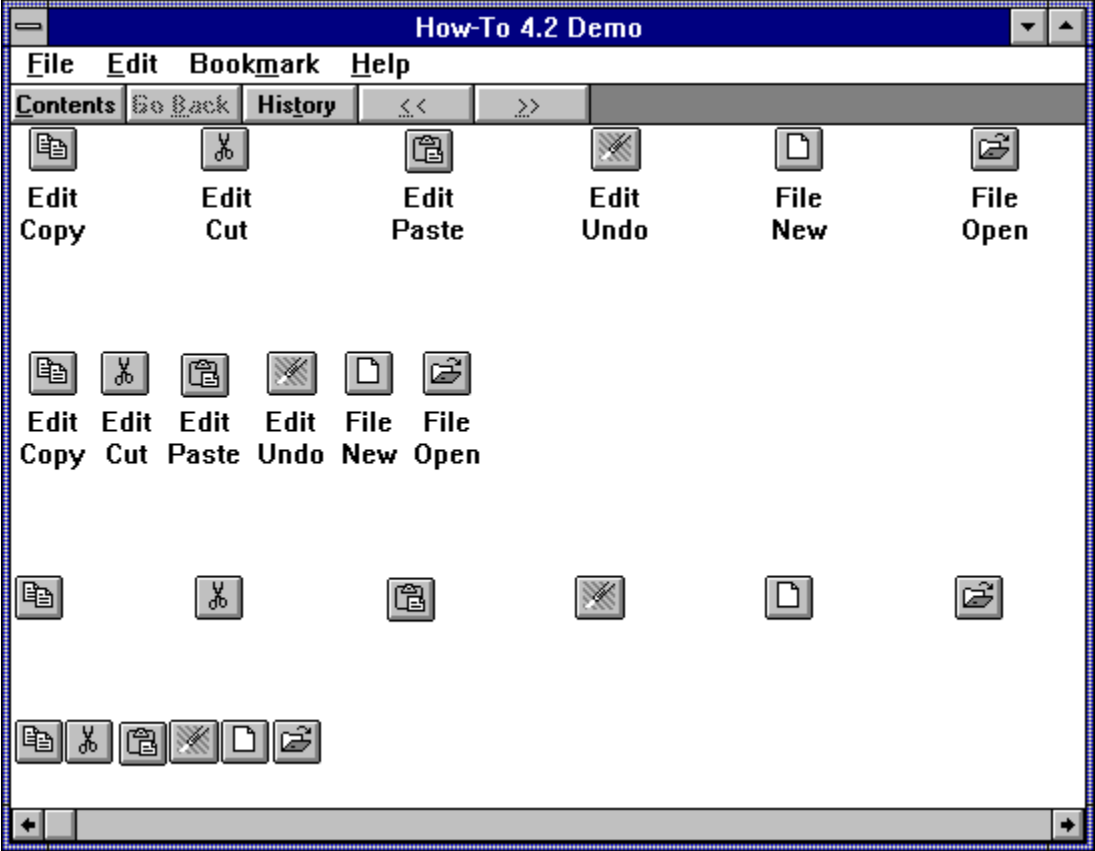


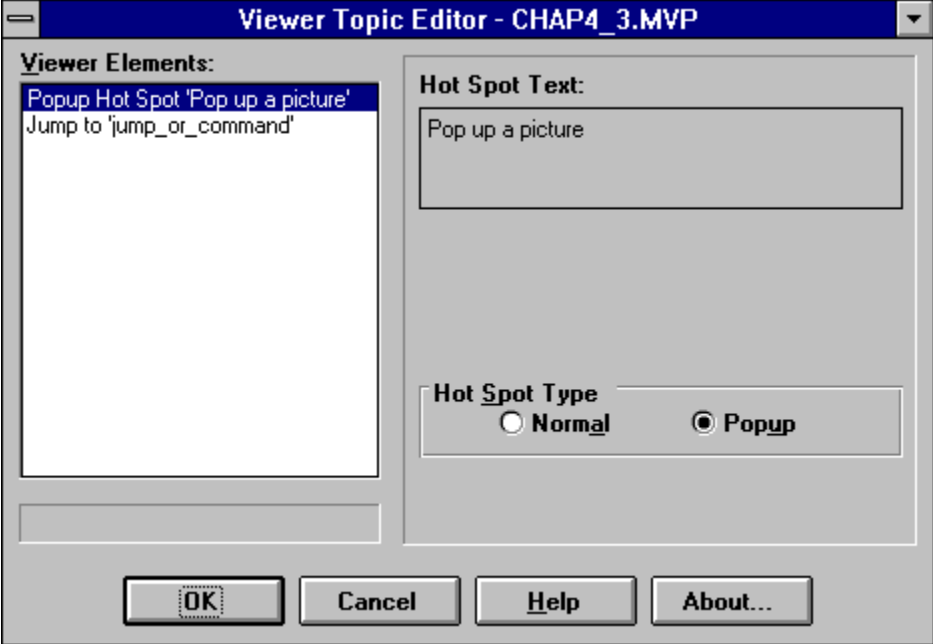
File  
New

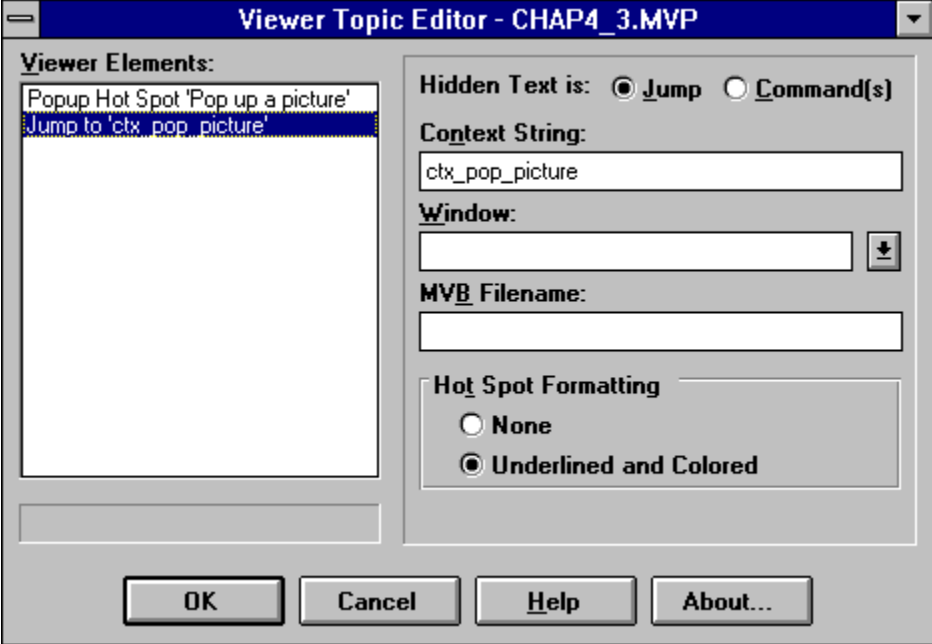


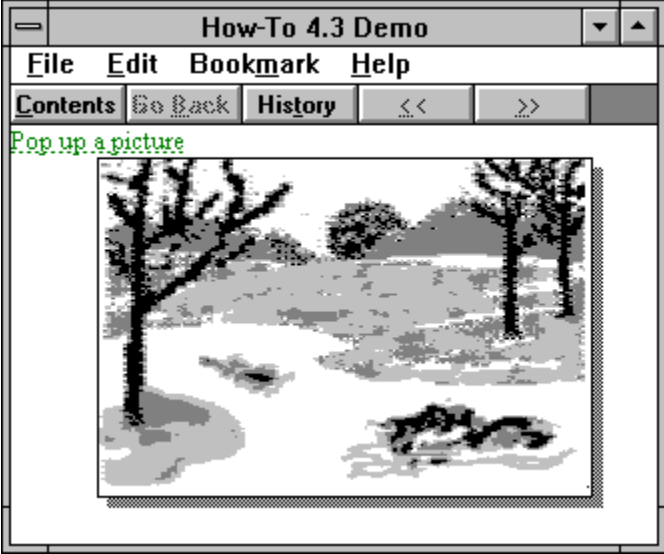
File  
Open











**Window Properties**

**Window Name:**

**Top:**       **Height:**

**Left:**       **Width:**

**Background Color:**  ...  **Use Default Color**

**Background Picture:**  ...

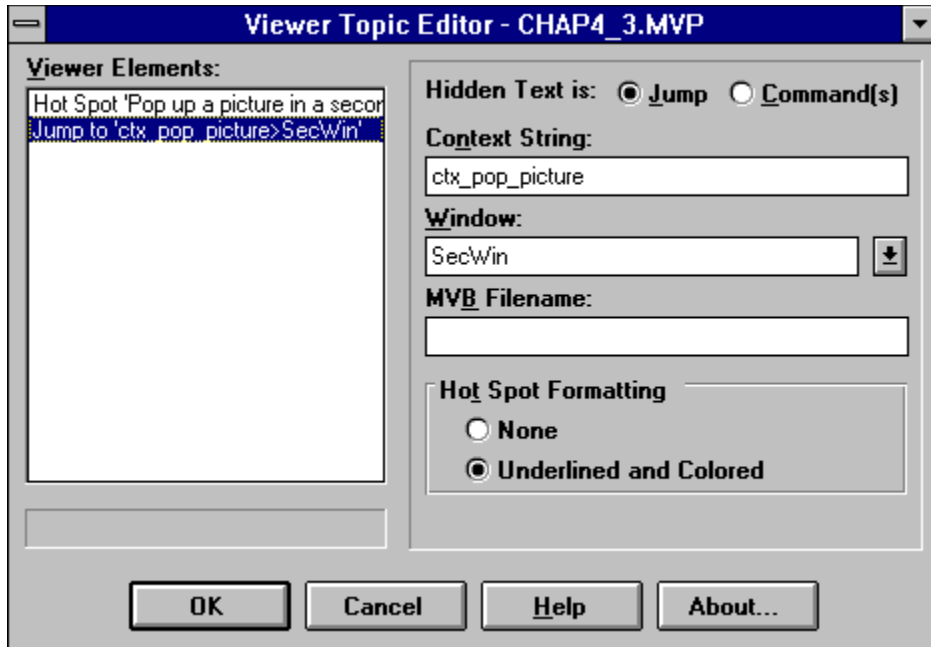
**Window Caption:**

**Initial State:**  ↓

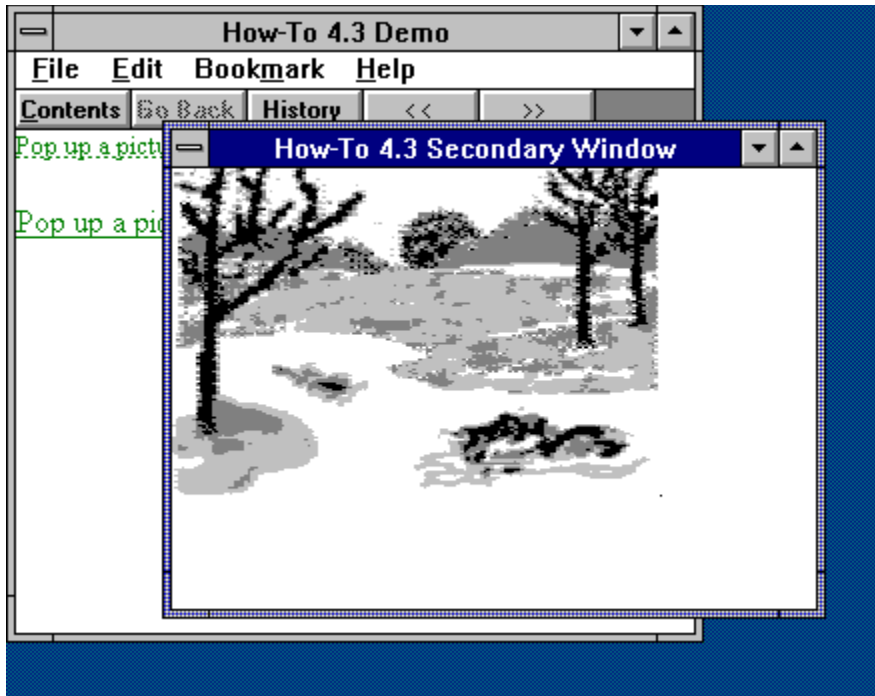
**Stay On Top:**  ↓

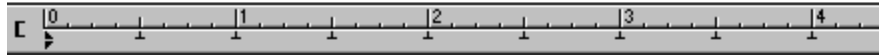
**Minimize with MAIN:**  ↓

**Coordinate System:**  ...









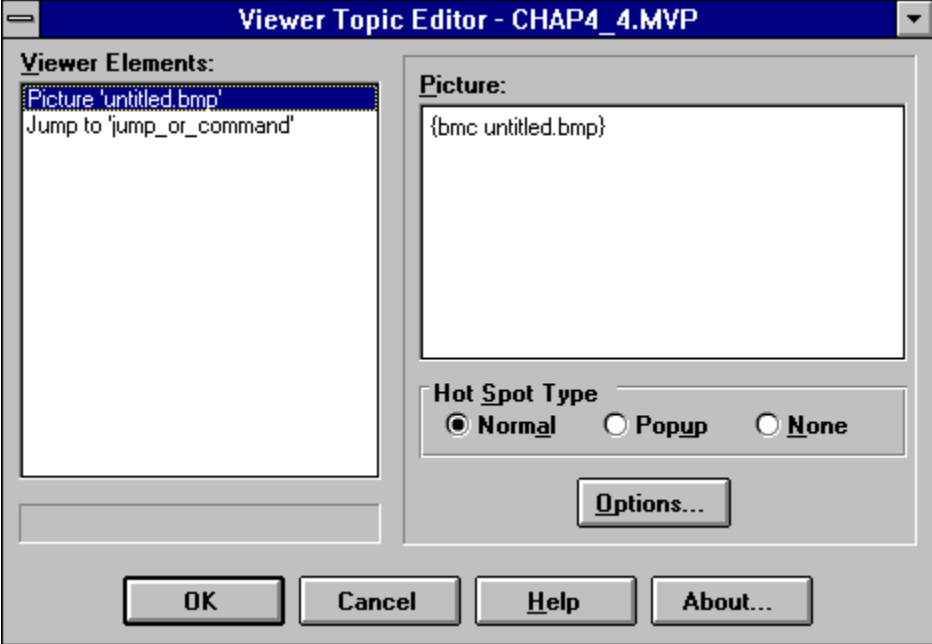
Pop-up-a-picturectx\_pop\_picture¶

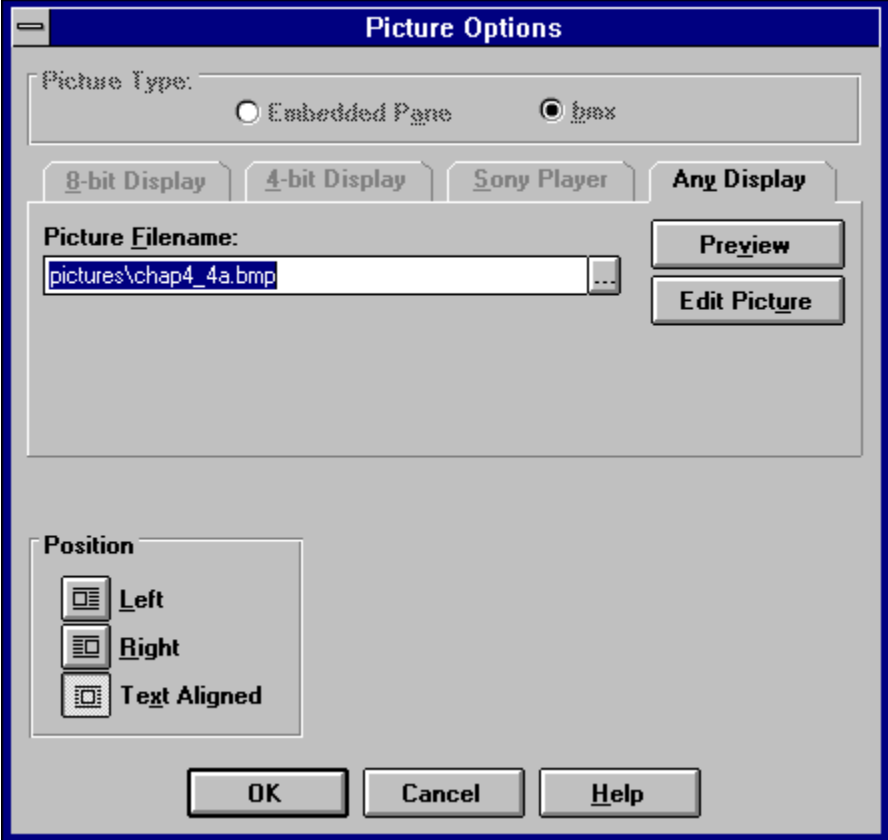
¶

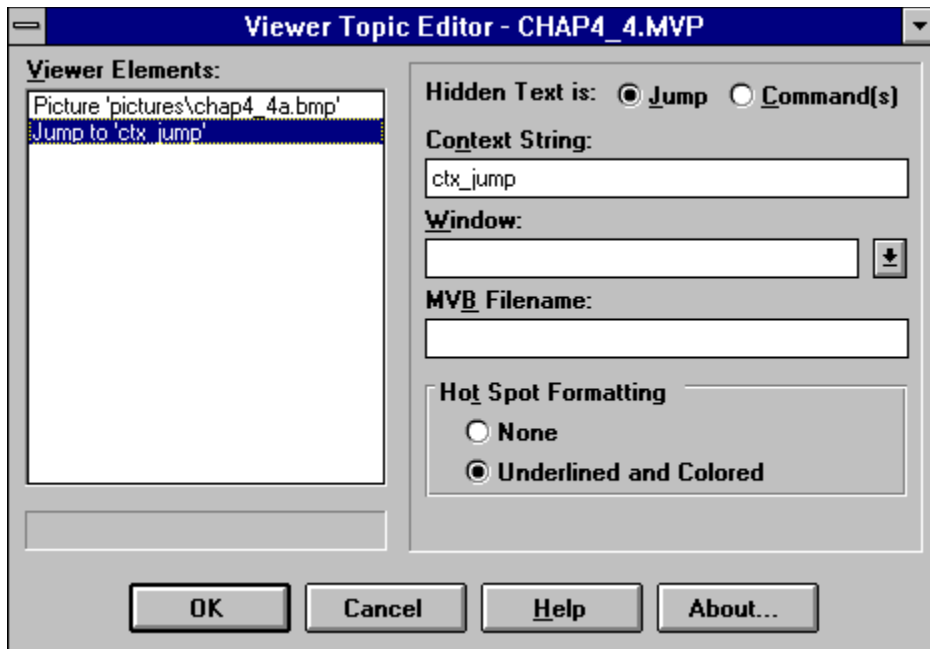
Pop-up-a-picture-in-a-secondary-windowctx\_pop\_picture>SecWin.¶

---

#{ewc-MVBMP2,ViewerBmp2,!chap4\_3.bmp}.¶









**Edit Command**

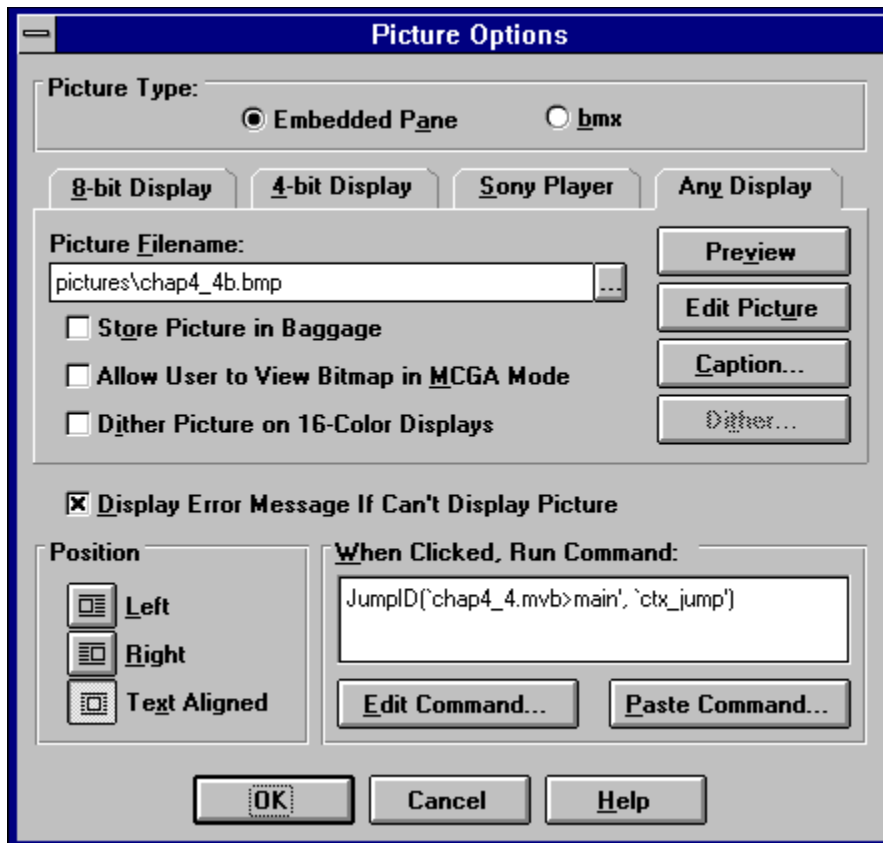
**Edit Command:** JumpID

**TitleFile:**

**WindowName:**

**Context:**

Context: Specifies the context string of a topic







```
{bmc:pictures\chap4_4a.bmp}ctx_jump¶
```

```
¶
```

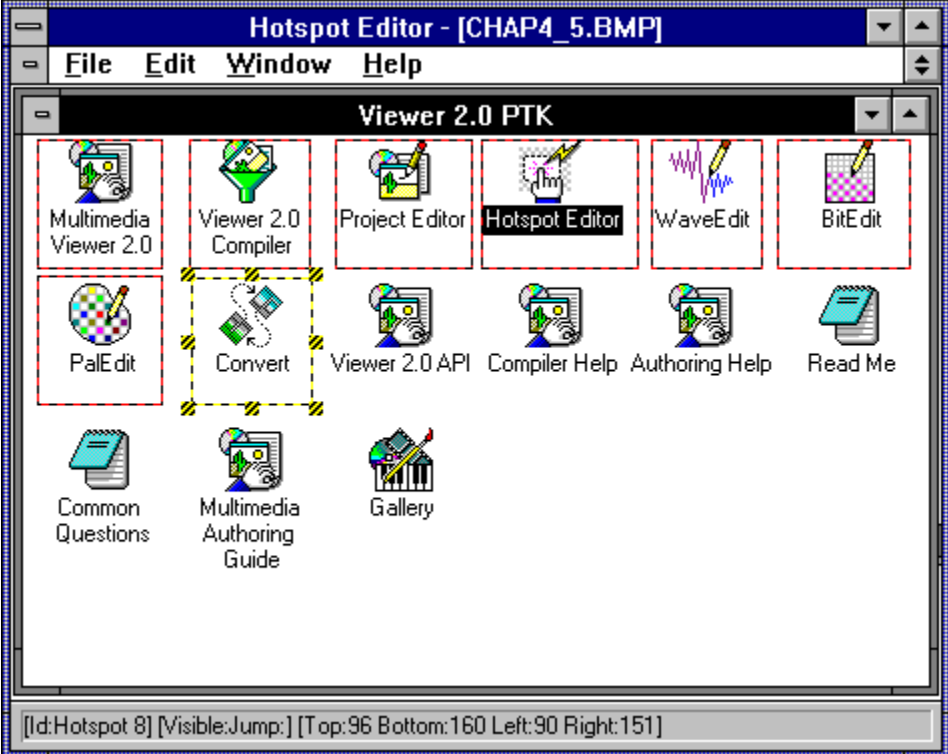
```
¶
```

```
{ewc:MVBMP2,ViewerBmp2,[macro="JumpID('chap4_4.mvb>main','ctx_jump)"]  
pictures\chap4_4b.bmp}¶
```

---

```
#This is the second topic.¶
```

```
—
```



**Attributes**

**Binding**

Context String:

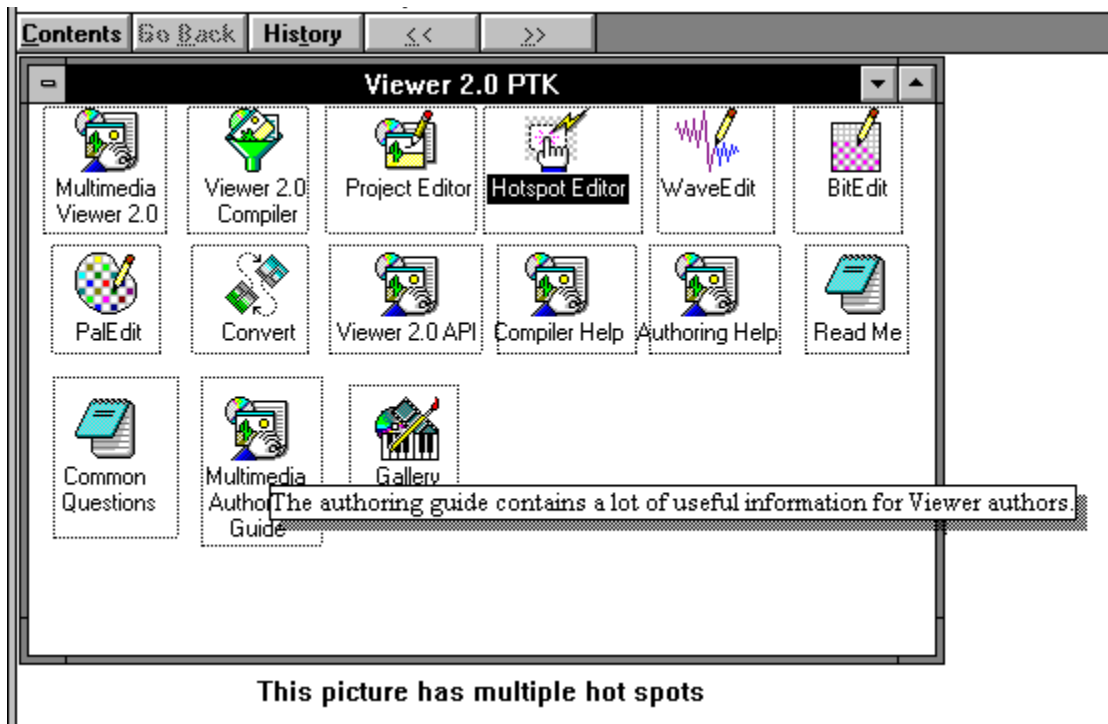
Type:   Attribute:

Hotspot Id:

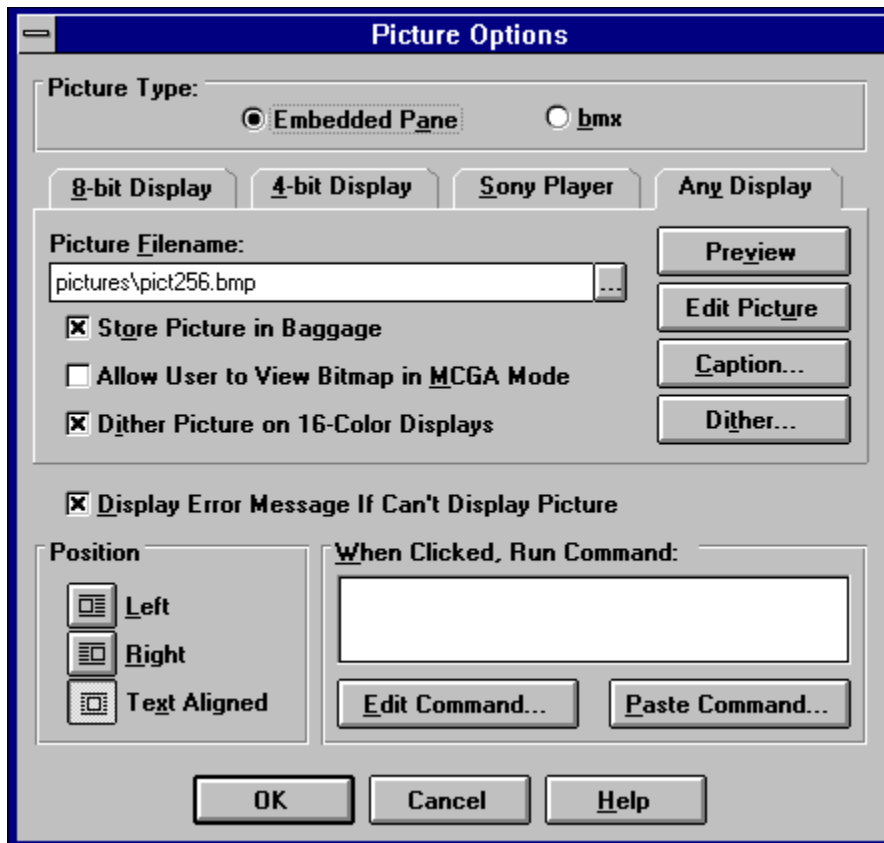
**Bounding Box**

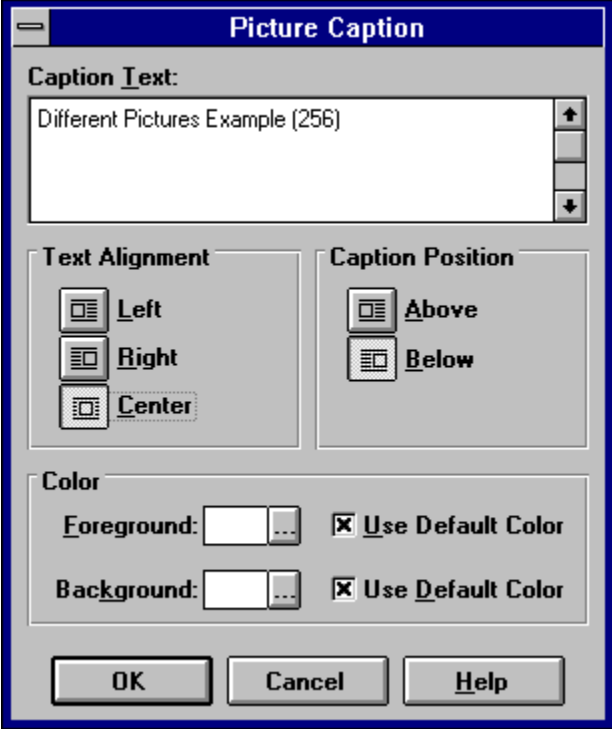
Left:  Top:

Right:  Bottom:



**This picture has multiple hot spots**







This chapter demonstrates various techniques for making your application more attractive or functional for the user. It also explains some of the possible problems or limitations of these techniques, to help you avoid problems.

This chapter shows you how to

- ü Use different fonts and colors to emphasize text, to change the appearance of hot spots, to change the colors of an entire pane or window
- ü Create a menu or toolbar item in Word to run the Windows CharMap program
- ü Use special characters to make your application even more interesting by using the extended characters from standard fonts, and the characters from fonts such as Symbol and Wingdings
- ü Use window panes to display several related parts of your application together
- ü Use a custom popup pane to control the size and location of popup windows
- ü Use secondary windows to display information that remains visible until the user closes its window
- ü Use a non-scrolling region to display text or other material that must stay in place as you scroll through the rest of the topic
- ü Control word wrap to display a wide section of text that must retain its formatting, using a horizontal scroll bar instead of wrapping the text to fit the space available
- ü Use tables to display information in rows and columns, and work around the limited formatting options
- ü Create a Welcome... topic to display information only when the application is started
- ü Start another program as the result of an action by the user



## Use Different {vflid137438953482}Fonts{vflid11132555231232} and {vflid137438953482}Colors{vflid3800470531342336}?

Complexity: EASY

### Problem

I want to use different fonts and colors to emphasize portions of my text, including choosing a unique appearance for hot spots.

### Technique

You use standard features of Word to select various fonts and colors. You use a combination of Word features and Topic Editor options to do the same thing in hot spots. You also demonstrate how you can select different colors for the background of selected windows or panes.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to Sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP5\_1, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP5\_1. Create these four directories for all projects, even if some are not needed.
2. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP5\_1 directory. Enter the name of your document file: TEXT\CHAP5\_1.RTF.
3. Use Project Editor to start Word and create your document.

Now you can proceed with this How-To. First, see how various common fonts look in Viewer:

4. Choose the Times New Roman font using Word's ribbon, and enter a line of text in your document:

This is Times New Roman

5. Click on the Bold button on the ribbon, and type the line below. Click on the button again to deselect Bold.

This is Times New Roman Bold

6. Repeat steps 4 and 5 for the Arial and Script fonts.

7. Select the Times New Roman font again, and type

This is Symbol:

8. Without moving the insertion point, select the {vflid137438953482}Symbol{vflid-9223349496866406400} font and type the lowercase letters a through e. Press [ENTER] to go to the next line.

9. Select the Times New Roman font again, and type

This is Wingdings:

- Without moving the insertion point, select the {vfld137438953482} Wingdings {vfld-9223349496866406400} font and type the lowercase letters a through e. Press **[ENTER]** to go to the next line.

Now include some of the character formatting options, and see how they look:

- Select the Times New Roman font again, and type:  
This shows subscript and superscript.
- Select the word *subscript*, then choose Character from the Format menu. Click on the arrow next to Super/subscript to display the choices. Select Subscript, then click on OK.
- Repeat step 12, but apply Superscript formatting to the word *superscript*.
- On the next line, type the following text. Be sure to use lowercase, not capitals, in the body of the sentence.  
This shows small caps and all caps.
- Apply the small caps format to the words *small caps*.
- Apply the all caps format to the words *all caps*.
- On the next line, type:  
This is expanded spacing. This is condensed spacing.
- Apply the expanded and condensed spacing formats to the respective sentences. Your document should look like Figure 5–1.

```
{ewc vwrht2, TsTextButton, "Figure  
5i½1"[Macro=JI('viewerht.mvb>SecWin', `fig5_1')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

- Now save your document file, and compile and test the application as usual. (This procedure is described in detail in How-To 3.2.) You should get a compiler warning, *Warning 6176: No words in titled topics found to index; .MVB file will lack full-text index*. Ignore this for now.
- Your Viewer window should look like Figure 5–2. Notice that the words {vfld137438953484}subscript{vfld13331578486784} and {vfld137438953484}superscript{vfld13331578486784} appear even with the rest of the line, the term {vfld2305858952132296716} *all caps*{vfld2305858952132296716} is in lowercase, and the {vfld137438953484}expanded{vfld13331578486784} and {vfld137438953484}condensed{vfld8026821369691897856} spacing sentences don't look any different. These options are not supported by Viewer.

```
{ewc vwrht2, TsTextButton, "Figure  
5i½2"[Macro=JI('viewerht.mvb>SecWin', `fig5_2')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Next demonstrate how to create different appearances for hot spots:

21. Double-click on the name of your document file to start Word again for your document.
22. Drop down a couple of lines below the current text.
23. Choose Character from Word's Format menu, and set the following options: Font—Times New Roman, Points—15, Style—Bold & Italic. Click on OK.
24. Type the text:  
This is a hot spot
25. Select the text you just typed and call up Topic Editor with your hotkey.
26. Select Hot spot (text) and click on OK.
27. Select the Jump to ... entry.
28. Enter ctx\_dummy as the context string.
29. Select None as the Hot Spot Formatting. The dialog box should look like Figure 5-3.

```
{ewc vwrht2, TsTextButton, "Figure  
5i½3"[Macro=JI('viewerht.mvb>SecWin', `fig5_3')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

30. Click on OK to return to Word.
31. Insert a blank line, and set the character format to Arial Font, 15 Points, no Style selected, and Red Color. Click on OK.
32. Type the text:  
This is another hot spot
33. Repeat steps 25 through 30.
34. Create a new topic with the context string ctx\_dummy, and type in the following text. The document should look like Figure 5-4.  
This is the test topic.

```
{ewc vwrht2, TsTextButton, "Figure  
5i½4"[Macro=JI('viewerht.mvb>SecWin', `fig5_4')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

35. Save the document as usual and compile your application.
36. Test your application. You see that the two hot spots have only the format features you selected. They are not green or underlined, like normal hot spots.

Finally, demonstrate changing the colors for the entire window:

37. In the Project Editor, choose Window Definitions from the Section menu.
38. Click on the Properties button to display the characteristics of the main

window.

39. Click on the Master Pane button to display the Master Pane Properties dialog box.
40. Click on the ellipsis (...) next to {vfld137438953482}Background Color{vfld-8970462913399619584}, then select a color that is distinct from your normal background.
41. Use the System Control Box to close the Properties window, then click on OK to return to the main Project Editor window.
42. Save the project file, and compile and test your application.
43. You should see the same window as you had after step 36, but with the new background color you selected.

### **How It Works**

You can use any {vfld137438953484}font{vfld-9042102693018992640} in a Viewer application, as long as you are certain that the identical font is installed on every system where your application is used. If there is no font with the same name, Viewer selects the closest ANSI font—which probably won't look like you intended!

The warning message generated by the compiler in step 19 ({vfld137438953484}Warning 6176{vfld-8935279640822415360}) was caused by not defining a title in this topic. Viewer provides full-text search only for the text in topics with titles. This warning indicates that there weren't any topics with titles, so there won't be any full-text search.

The author-defined hot spots have a percent sign (%) at the beginning of the hidden text. This is the flag to the Viewer compiler that the standard green underlined format should not be applied. That flag was generated when you selected None for the Hot Spot Formatting option in Topic Editor.

The {vfld137438953484}background color{vfld-8935142201868943360} for the window was actually set for the Master Pane because that pane covered the entire window. If multiple panes are used, different background colors can be set for each. The background color of any areas in the Viewer window that are not covered by any panes can be controlled through the Window Properties dialog box. This can be important when panes are resized to fit topics that are smaller than the defined size of the pane—this exposes the window area below. This is also important if you define panes that do not cover the entire window surface at their full size.

### **Comment**

Viewer provides far broader font support than Windows Help, which only permits ANSI fonts and the Symbol font. The requirement that the fonts you use are present on the users' systems means that you should use only fonts distributed with Windows 3.1, unless you really need other fonts and can control the user's environment. Distributing fonts with your application may be legally restricted, and requires additional effort in the installation program.

Authors have reported finding more than one font called “{vfld137438953484}Dingbats{vfld-8935279640822415360},” each with different characters. This can cause unexpected characters to appear in your application.

Resist the urge to include many different fonts in your application. This results in an appearance known as the “ransom note effect” because of the similarity to a note made from words cut out of different parts of a magazine. Many authors find that two fonts, such as Arial and Times New Roman, with their bold and italic variants and different sizes, provide all the type styles they need.

The Viewer feature limiting full-text search to topics with titles is important—this is the only way you can control which text is searchable. You can prevent popup windows or other special-purpose topics from being displayed by the search function. This is why such topics usually are not defined with titles. The search and related functions are explained in detail in Chapter 11.

Support for author-defined hot spot appearance can be useful, but should be used carefully. You have to be consistent, and should explain your standard to the user at the beginning of the application. This support does not provide for underlining or any character formatting that Viewer doesn’t otherwise support.

Viewer does not support superscript, subscript, all caps, or expanded or condensed spacing character formatting. `{vfld137438953484}Strikethrough{vfld-8935142201868943360}` and hidden formats have special meanings in Viewer—they are used for hot spots—therefore they aren’t supported either. If you need superscript or subscript text in your application, the easiest way is to create the needed text in the Microsoft Equation application, then convert the results to a bitmap that can be included. Similar techniques can be used for expanded or compressed spacing. It is difficult to make these bitmaps line up properly with text. They look best on lines by themselves.

Microsoft warns that author-defined hot spot colors may be hard to read on VGA `{vfld137438953484}gas plasma{vfld12232066859008}` screens. Users can cause author-defined colors to be ignored by adding the line `{vfld2305865549202063371}Colors=NONE{vfld-9079242005371944960}` to the [Multimedia Viewer] section of their WIN.INI file. If you use nonstandard hot spots you should inform your users how to do this, just in case they have a problem.

The color of hot spots can also be controlled by adding or changing options in the user’s `{vfld137438953482}WIN.INI{vfld-9223349496866406400}` file. This should only be used in special circumstances, because it affects the appearance of every Viewer application. These options are placed in the [Multimedia Viewer] section of the WIN.INI file, in the form of `command=(RRR, GGG, BBB)` where RRR, GGG, BBB are the red, green, and blue components of the desired color. Each value can be in the range 0–255. The options are shown in Table 5–1.

**Table 5–1. WIN.INI Options**

|           | Command  | Default | Applies to ... |
|-----------|--|---------|----------------|
| (0,255,0) | <code>{vfld137438953483}JumpColor{vfld-8970462909104652288}</code><br>Jump Hot Spots             |         | Green          |
| JumpColor | <code>{vfld137438953483}IFJumpColor{vfld-8970462909104652288}</code><br>Interfile Jump Hot Spots |         | Same as        |

|           |   |         |
|-----------|---|---------|
| JumpColor | {vfld137438953483}PopUpColor{vfld-8970462909104652288}<br>Popup Hot Spots           | Same as |
| JumpColor | {vfld137438953483}IFPopUpColor{vfld-35321248401588224}<br>Interfile Popup Hot Spots | Same as |
| JumpColor | {vfld137438953483}MacroColor{vfld-35321248401588224}<br>Macro Command Hot Spots     | Same as |

---

## Run CharMap from Within Word?

Complexity: INTERMEDIATE

### Problem

I use various special characters frequently. I want to be able to run Windows CharMap easily from Word to help me select the characters.

### Technique

You create a simple macro that runs CharMap, then add it to Word's menu or toolbar.

This How-To does not require creating any new directories, project files or document files. It updates the NORMAL.DOT template file in your WINWORD directory. No sample files are provided on the CD disk because this section requires choices of personal preference.

### Steps

First you *must* make a safety copy of your current NORMAL.DOT file:

1. Start Windows File Manager, and select the directory where Word is installed on your system. This is usually named WINWORD.
2. Select the file NORMAL.DOT.
3. Choose Copy from the File menu.
4. Enter NORMAL.SAV in the To field and click on OK.

Now you can create a Word macro that starts the program:

5. Start Word. You don't need to load a document or start a new one. Word automatically starts a document named Document1.
6. Choose Macro from the Tools menu.
7. Enter CharMap in the Macro Name field. Enter Start CharMap in the Description field.
8. Click on the Edit button. Word displays a document window containing the lines "Sub MAIN" and "End Sub" separated by a blank line.
9. On the blank line enter the following line, as shown in Figure 5-5.  
Shell "charmap", 1

```
{ewc vwrht2, TsTextButton, "Figure  
5½"[Macro=JI('viewerht.mvb>SecWin', `fig5_5')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

10. Choose Close from the File menu. Your macro is now created.
11. To test your macro, choose Macro from the Tools menu, then select the CharMap macro and click on the Run button.

Next add the macro to your menu. If you want to add it to your toolbar instead, skip to step 19. You can add it to both if you want.

12. Choose Options from the Tools menu.
13. Scroll down to the Menus category and click on its icon.
14. Under the Menu heading select the T&ools entry, and add your new item under that menu heading. Use a different one if you prefer—the rest of the steps are the same.
15. Under the Macros heading select the first entry, with a series of dashes, and click on the Add button. This inserts a separator line at the bottom of the menu list.
16. Under Macros select the CharMap entry.
17. Change the Menu Text field from &Char Map to Char Ma&p. The letter following the ampersand becomes the underlined letter that can be used to select the menu item through the keyboard. This dialog box defaults to using the first letter, but doesn't check for conflicts. Since *C* is already used for Calculate, use *p*.
18. Click on the Add button, then click on the Close button.

To add your macro to the toolbar:

19. Choose Options from the Tools menu.
20. Scroll to the toolbar category and click on its icon to select it.
21. Your current buttons, and the spaces between them, are listed under the Tool to Change heading. Click on the down arrow to see the list. Select a button to replace with your new CharMap button. You might consider the other application buttons such as InsertChart. Be sure to select a button you don't use often!
22. Scroll through the list of icons under the Button heading, and select one you want to represent CharMap. Be sure you don't pick one you are already using. That would be confusing. My personal choice is the icon listed after the Bulleted List icon and before the Unindent icon.
23. Select the CharMap entry under the Macros heading.
24. Click on the Change button to make your changes take effect, then the Close button.

After adding your macro to either or both places:

25. Exit Word as usual. Word displays a message box asking *Do you want to save the global glossary and command changes?*
26. Click on the Yes button to save the changes you just made.

### **How It Works**

The one-line macro starts CharMap. Because it's the latest program to start, this window displays on top of Word.

The next How-To demonstrates the use of CharMap.

### **Comment**

The shell command in the macro can be used to start any program, using the same syntax that can be used by choosing Run from the Program Manager's



File menu. The command can refer to a Windows or DOS executable file, a PIF file, or a data file with an extension that is associated with a program. The directory path is required if the program or data file cannot be located by Windows.

The same technique can be used to make it easy to start other programs you use frequently while in Word, such as Notepad or a drawing or painting program.

The techniques used in this How-To are explained in more detail in the Word documentation. Separate documentation of WordBasic, the macro language, is also available.

If you want to undo your changes, you can use File Manager to delete your new NORMAL.DOT file and rename NORMAL.SAV to NORMAL.DOT. This restores your previous macros, menus, and toolbar. You could also choose Options from the Tools menu to reverse the effect of your changes through steps similar to those above. When making global changes such as these you should always plan how to undo them, by copying the file or recording the back-out steps that would be required.

## Use Special Characters?

Complexity: INTERMEDIATE

### Problem

I want to use various special characters, including some from the Symbol and Wingdings fonts.

### Technique

You use Windows CharMap to select the desired characters, and the standard Windows technique for inserting them into your document. You also see which characters need some extra effort to be used.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to Sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP5\_3, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP5\_3. Create these four directories for all projects, even if some are not needed.
2. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP5\_3 directory. Enter the name of your document file as TEXT\CHAP5\_3.RTF.
3. Use the Project Editor to start Word and create your document.

Next use a few special characters and see what happens:

4. Select the Times New Roman font.
5. Enter this text in your document:

This is a copyright symbol:

6. Start Windows CharMap. If you did How-To 5.2, use the menu item or toolbar icon you created. Otherwise use Program Manager by choosing Run from the File menu and entering Charmap as the program to be run.
7. Select the Times New Roman font to be displayed.
8. Locate the copyright symbol. It is three rows below the capital *I*.
9. Click on that symbol. An enlarged view displays to help you make sure you selected the right character, and the keyboard code shows in the lower-right corner of the window as "Alt+0169."
10. Click on the Close button and return to Word. The insertion point should still be immediately after the text you entered. If not, put it there.
11. If NumLock is off, turn it on by pressing the NumLock key.
12. Hold down **[Alt]** and type the digits **0169** on the numeric keypad on the right side of your keyboard. The copyright symbol should appear in your

document.

13. On the next line type:  
These are from the Symbol font:

14. Start CharMap again, and select the Symbol font.
15. Locate the ® and © symbols. Note that there are two of each, one on the last line and one on the line above.
16. Insert all four symbols in your document, using the same procedure as in step 12. Be sure to select the Symbol font.
17. Select the Times New Roman font again, drop down another line and type:

These are single quotes:

18. Enter the quotes as [ALT]-0145 and [ALT]-0146. Your document should look like Figure 5-6.

```
{ewc vwrht2, TsTextButton, "Figure  
5i½6"[Macro=JI('viewerht.mvb>SecWin', `fig5_6')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

19. Save your document and compile and test it as usual. It should look like Figure 5-7.

```
{ewc vwrht2, TsTextButton, "Figure  
5i½7"[Macro=JI('viewerht.mvb>SecWin', `fig5_7')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

20. Everything looks the same, except the quote marks are missing! Word uses special codes in the RTF file to represent certain characters, including quotation marks. Viewer does not recognize these codes, so the characters are ignored. Fortunately, you can fix this.
21. Start Notepad and load your document file: \VIEWERHT\CHAP5\_3\TEXT\CHAP5\_3.RTF. The file should look like Figure 5-8.

```
{ewc vwrht2, TsTextButton, "Figure  
5i½8"[Macro=JI('viewerht.mvb>SecWin', `fig5_8')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

22. Near the end of the file you see your text These are single quotes. At the end of that line you find the codes \lquote and \rquote. These are the codes that Viewer can't recognize.
23. Replace \lquote with \'91. The quote mark before the 91 is the standard right single quotation mark, on the same key as the double quotation mark. Be sure to remove the blank that followed the \lquote code—this space is part of the code.
24. Replace \rquote with \'92 similarly.

25. Save the file, exit Notepad, and use the Project Editor to compile and test your application. The quotation marks should now be present!

**How It Works**

Windows allows you to enter any character, even if it is not represented on the keyboard, by holding [ALT] while entering the four-digit code for the desired character. This code always starts with a zero that must be entered. The numbers must be entered on the numeric keypad while NumLock is on. CharMap makes it easy to locate the desired characters and find their keyboard codes.

Note that the codes for common characters are not always consistent among fonts. The copyright symbols used here demonstrate this fact. This is part of the reason why you should only use fonts that are certain to be present on the users' systems.

**Comment**

Viewer cannot use the RTF codes generated by Word for single and double quotation marks, bullets, and em- and en-dashes. The Word codes and the replacements needed for Viewer are shown in Table 5-2. Note that the codes used by Word always include a single trailing blank that must be replaced as part of the code.

**Table 5-2. Viewer RTF Code Replacements**

| Typographic Code  | Word Code  | Viewer Code |
|---|------------|-------------|
| Open {vfld137438953482}single quotation mark {vfld-8970462909104652288} (‘)   | \quote     | \91         |
| Close {vfld137438953484}single quotation mark {vfld-8970462909104652288} (’)  | \rquote    | \92         |
| Open {vfld137438953482}double quotation mark {vfld-8970462909104652288} (“)   | \dblquote  | \93         |
| Close {vfld137438953484}double quotation mark {vfld-8970462909104652288} (”)  | \rdblquote | \94         |
| Bullet (·)\bullet\95 {vfld137438953484} {vfld-9223339601261756416}En dash (—) | \endash    |             |
| \96 erñ {vfld137438953484} {vfld-35321248401588224}                           |            |             |
| Em dash (—)\emdash\97 {vfld137438953484} {vfld-35321248401588224}             |            |             |

The codes for an en-dash and em-dash are documented incorrectly in Viewer's Authoring Guide. They are correct here and in the Viewer Technical Reference.

These replacements can be made in Word if you did not suppress the File Conversion dialog box. When loading the file, change the format in that dialog box from RTF to Text Only. This leaves the RTF commands visible to

be updated. You can then use Word's Search and Replace feature to make the necessary changes. Be sure to save the file in Text Only format, using the Save As menu item. The RTF files for real applications are normally far too large for Notepad.

**Use Window {vflid2305852355062530058}Panels{vflid-9079245303906828288}?**

Complexity: INTERMEDIATE

**Problem**

I want to display a picture and several sections of text at once, and keep them coordinated.

**Technique**

You use the Project Editor to define two different pairs of regular panes, and reduce the size of the master pane accordingly. You use topic entry commands to cause the proper topics to be displayed.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to Sections 3.1 and 3.2.)

**Steps**

First you must create the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP5\_4, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP5\_4. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the picture files for this How-To from the VIEWERHT\HOWTOS\CHAP5\CHAP5\_4\PICTURES directory on the CD-ROM to the VIEWERHT\CHAP5\_4\PICTURES subdirectory on your hard drive.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP5\_4 directory. Enter the name of your document file as TEXT\CHAP5\_4.RTF.

Now you can proceed with this How-To:

4. Choose Window Definitions from the Section menu.
5. Click on the Properties button to display the main window properties.
6. Clear the Auto-Position check box. Click on the Master Pane button to display the Master Pane Properties dialog box. Clear the Auto-Position check box in this dialog box as well. Click on the Preview On button to display the window.
7. Click in the master pane and the resizing handles appear. Drag the handles so that the master pane covers an area in the lower half of the center of the window, as shown in Figure 5-9.

```
{ewc vwrht2, TsTextButton, "Figure
5½9"[Macro=JI('viewerht.mvb>SecWin', 'fig5_9')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

8. Close the window by using the System Control Box at the upper left

- corner. Close the Window Properties dialog box the same way.
9. Click on the Panes file folder tab in the Window Definitions dialog box, then click on the New button to define the first new pane. It is labelled 'Pane1'.
  10. Click on the Properties button. Change the Pane Name to Pane1A and change the Border to (None). Leave the Dismiss When field set to "When topic in master pane changes."
  11. Click on the Windows button to display the Pane Associations dialog box.
  12. Click on the Show in Window check box to set it, then click on the Preview button to display the window.
  13. Drag the pane so it covers the area to the left of the master pane. Close the window as in step 8.
  14. Repeat steps 9 through 13 to create pane PANE1B on the right side of the master pane.
  15. Repeat steps 9 through 13 to create PANE2A, covering from the upper left corner of the window to the upper right corner of the master pane.
  16. Repeat steps 9 through 13 to create PANE2B, covering the area to the right of the master pane, from the top to the bottom of the window. The window should look like Figure 5–10 when you are done. Note that the title of PANE1B is covered by PANE2B.

{ewc vwrht2, TsTextButton, "Figure  
5i½10"[Macro=JI('viewerht.mvb>SecWin', `fig5\_10')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}

17. Click on OK in Windows Definitions to return to the main Project Editor window.
18. Choose Title Options from the Section menu and enter contents as the Contents Topic. Click on OK.
19. Choose Groups from the Section menu to define a topic group.
20. Click on the New button to create a new group. Click on the Searchable check box to clear it. This group is used only for a browse sequence, not as a search group. Click on OK.
21. Save your project file, then use the Project Editor to start Word and create your document.
22. Use Topic Editor to set the context string to contents.
23. Use Topic Editor to set the Topic Browse Sequence to Group1 and Browse sequence to 010. (Refer to How–To 3.3 if you are not familiar with this procedure.) The dialog box should look like Figure 5–11 when you are done.

{ewc vwrht2, TsTextButton, "Figure  
5i½11"[Macro=JI('viewerht.mvb>SecWin', `fig5\_11')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}

24. Bring up Topic Editor and select the Topic Entry command. Click on the Paste Command button and select the {vfld137438953483}PaneID{vfld1477325272146509824} command.
25. Click on the Edit Command button, and set the options as follows. Change Title File to qchPath (without quotes), WindowName to one space (without quotes), Context to `topic1a`, Pane Name to `Pane1a`, and PrintTabCopyOrder to 0. (Refer to How-To 3.6 if this procedure is unfamiliar.) Click on OK.
26. Repeat steps 24 and 25 for context string `topic1b` in pane `Pane1b`.
27. Enter the following text:  
This is the first topic. It is displayed in the master pane.
28. Create a new topic, with context string topic1a. Enter the following text:  
This is part of the first topic. It is displayed in regular pane Pane1A.
29. Create a new topic, with context string topic1b. Use Topic Editor to insert a picture command with file name PICTURES\CHAP5\_4A.BMP, check the Store in Baggage checkbox, and enter the following text as the caption.  
This picture goes with the first topic.[ENTER]It is displayed in regular pane Pane1B.
30. Repeat steps 22 through 29 to create similar topics with context strings topic2, topic2a, and topic2b. Refer to panes Pane2A and Pane2B, and picture file PICTURES\CHAP5\_4B.BMP.
31. Save the document file, compile, and test as usual.
32. Use the Browse buttons to display the two sets of topics. They should look like Figures 5-12 and 5-13.

```
{ewc vwrht2, TsTextButton, "Figure
5i½12"[Macro=JI('viewerht.mvb>SecWin', `fig5_12')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

```
{ewc vwrht2, TsTextButton, "Figure
5i½13"[Macro=JI('viewerht.mvb>SecWin', `fig5_13')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

The first topic displays in the master pane by default, because you did not force anything else to happen. This topic contains topic entry commands that cause the other two topics to be displayed in their respective regular panes. When you browse to the other topic, Viewer immediately clears the regular panes associated with the previous topic because the panes were defined with “Dismiss when topic in master pane changes.” The new topic displayed by the browse sequence immediately executes its topic entry commands, displaying the associated topics in their panes.



The value in the PrintTabCopyOrder field of the PaneID command determines the sequence that panes are printed or copied. A value of zero prevents them from being either printed or copied.

### **Comment**

There can only be one master pane. It remains in a fixed location in a fixed size unless a {vfld137438953483}PositionMaster{vfld12232066859008} command is executed. Commands {vfld2305858952132296715}MasterAspect{vfld2305857852620668928}, {vfld2305858952132296715}MasterNSRColor{vfld2305857852620668928}, and {vfld2305858952132296715}MasterSRColor{vfld2305857852620668928} can be executed to change the appearance or colors in the {vfld137438953482}master pane {vfld2378181537062453248}. These commands are explained in the Viewer Authoring Help file installed on your system from the enclosed CD-ROM disk. These commands usually are executed as part of a topic entry if they are needed. As shown in this How-To, there can be many regular panes of assorted sizes and positions.

Notice that the text below the picture is {vfld137438953484}cut off{vfld4081945508052992} in Figure 5-12. Regular panes cannot show contents that are larger than the pane, and don't show scroll bars. Any excess is simply cut off, as shown.

When a list box is displayed, such as in the Paste Command operation, you can jump to the desired part of the list by pressing the key for the first letter of the desired command. For example, pressing *p* takes you directly to the PaneID command.

When defining multiple topic entry commands, you can create each one individually or combine them. To combine several commands in one entry start by defining the first as usual. After the command has been edited, while in the Topic Editor dialog box, position the insertion point at the beginning of the next line, then repeat the process of Paste Command and Edit Command.

## Use a Custom Popup Pane?

Complexity: INTERMEDIATE

### Problem

I want to define a popup window, but I need to control its size and position.

### Technique

You define a custom popup in Project Editor, and force the topic to be displayed in that popup.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to Sections 3.1 and 3.2.)

### Steps

First you must create the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP5\_5, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP5\_5. Create these four directories for all projects, even if some are not needed.
2. Use Project Editor to create a new project file in your VIEWERHT\CHAP5\_5 directory. Enter the name of your document file as TEXT\CHAP5\_5.RTF.

Next define the custom popup window:

3. Choose Window Definitions from the Section menu.
4. Click on the Popups file folder tab, then click on the New button to define a new popup window. Leave the default name, Popup1, unchanged.
5. Click on the Properties button to view the window size, location and colors. Leave all of these unchanged for now. Changes you might make in a real application are explained in the Comment section at the end of this section. Use the System Control Box at the upper left corner of the window to return to the Window Definitions window. Click on OK to return to the main Project Editor window.

Next create the document file to test the popup:

6. Save your project file, then use Project Editor to start Word and create your document.
7. Enter the text:

Display a custom popup window

8. Select the text you just entered and call up Topic Editor.
9. Select the Hot spot (text) entry and click on OK.
10. Leave the Hot Spot Type as Normal. Custom popups are not defined as Hot Spot Type Popup.

11. Select the Jump to... element, and click on Hidden Text is command.
12. Click on the Paste Command button and select the {vfld137438953483}PopupID{vfld1477325272146509824} command.
13. Click on the Edit Command button. Enter qchPath in the Title File field and `ctx\_popup' as the Context. Click on the arrow next to the Popup Name field, and select Popup1. Click on OK twice to return to your document.
14. Use Topic Editor to create a new topic with context string ctx\_popup. Enter the text:  
This is the custom popup
15. Save the document file and compile and test your application as usual. The result should look like Figure 5–14.

```
{ewc vwrht2, TsTextButton, "Figure  
5½14"[Macro=JI('viewerht.mvb>SecWin', `fig5_14')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

The PopupID command is executed when the user clicks on the hot spot. This command causes the specified topic to be displayed in the custom popup window named in the command. All characteristics of the window were defined through Project Editor dialog boxes.

The custom popup window is removed when the user clicks anywhere on the screen, just as with standard popup windows.

### Comment

The location and size of the custom popup window are relative to the entire screen, and do not have to be located within the Viewer window at all. The corresponding characteristics of the Viewer window should be considered when setting these values. The custom popup definition is only used if the relative positions are critical—otherwise the much easier standard default popup is adequate. The size and position of the main window must be fixed to assure the desired relative positions.

## Use {vfld2305852355062530058}Secondary Windows{vfld-9079245303906828288}?

Complexity: EASY

### Problem

I want to display information in a window that can remain visible as long as the user wants it to.

### Technique

You define the secondary window through Project Editor, and use Topic Editor to create the command in the document file to display a topic in that window.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to Sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP5\_6 and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP5\_6. Create these four directories for all projects, even if some are not needed.
2. Use Project Editor to create a new project file in your VIEWERHT\CHAP5\_6 directory. Enter the name of your document file as TEXT\CHAP5\_6.RTF.

Next define the secondary window:

3. Choose Window Definitions from the Section menu.
4. Click on the New button to define a new window.
5. Click on the Properties button to view the window characteristics, as shown in Figure 5-15.

```
{ewc vwrht2, TsTextButton, "Figure
5i½15"[Macro=JI('viewerht.mvb>SecWin', 'fig5_15')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

6. Leave the properties unchanged for this How-To. In a real application you would set the size and location as needed for your material. Use the System Control Box to close this dialog box, and click on OK to return to the main Project Editor dialog box.
7. Save your project file by choosing Save from the File menu.

Next create the document:

8. Double-click on the name of your document file to start Word and create the file.
9. Enter the hot spot text:

## Display a secondary window

10. Select the text and call up Topic Editor. Select Hot Spot (text) and click on OK.
11. Select the Jump to... element.
12. Enter context string `ctx_second`. Click on the arrow alongside the Window field and select the secondary window (Win1). Click on OK to return to the document.
13. Create a new topic with context string `ctx_second`. Enter the text: This is being displayed in a secondary window.
14. Save the document file and compile and test the application as usual.
15. The result should look like Figure 5–16.

```
{ewc vwrht2, TsTextButton, "Figure  
5i½16"[Macro=JI('viewerht.mvb>SecWin', `fig5_16')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

16. The secondary window remains in place if you click on the screen. It can be minimized to an icon and restored. It can be closed through the System Control Box at the upper left corner of the window. It closes automatically when you close the main window.

### How It Works

Specifying a window name in the hot spot definition causes Topic Editor to create the hot spot for a secondary window by adding `>windowname` to the command. This form is only used for secondary windows.

### Comment

The window definition allows you to specify if the secondary window should be minimized when the main window is minimized. This is normally set to Yes to keep the windows coordinated. This would be changed if the secondary window should appear independent to the user.

All commands in the `{vfld137438953484}Config` section `{vfld7237002585041797120}` of the project file are executed every time a secondary window is displayed, as well as when the main window is displayed. These commands may perform functions that can be repeated harmlessly, such as defining buttons or menu items that are never changed later. The section could also include commands that force a particular topic to be displayed, play sound or movie files, or perform other actions that should not be repeated. The effect of this reexecution may force you to choose between special effects when the application is started or secondary windows. How-To 5.10 demonstrates a case where this is a problem, with a solution.

The secondary window uses the same `{vfld137438953482}icon` `{vfld7237002585041797120}` as the main window. The icon can be set through Project Editor, using the Title Options choice in the Section menu.

## Use a {vflid2305852355062530058}Non-Scrolling Region{vflid-9079245303906828288}?

Complexity: EASY

### Problem

I need to keep a section at the top of my topic visible as the user scrolls through the text in the rest of the topic. I would like to make this region stand out from the rest of the topic.

### Technique

A *non-scrolling region* (NSR) is specified through special paragraph formatting in the document. You also use a Viewer option to change the background color of this region.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to Sections 3.1 and 3.2.)

### Steps

First prepare the directories and files, and change some properties of the master pane:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP5\_7, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP5\_7. Create these four directories for all projects, even if some are not needed.
2. Use Project Editor to create a new project file in your VIEWERHT\CHAP5\_7 directory. Enter the name of your document file as TEXT\CHAP5\_7.RTF.
3. Reduce the size of the master pane, as demonstrated in How-To 5.4. This reduces the amount of text needed to cause scrolling.
4. While the Master Pane Properties dialog box is displayed, select a {vflid137438953482}background color {vflid71919613918576640} for the NSR, as shown in Figure 5-17. Return to the Project Editor main window as you did in previous sections, then save your project file.

{ewc vwrht2, TsTextButton, "Figure  
5i½17"[Macro=JI('viewerht.mvb>SecWin', `fig5\_17')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}

Next create the document file:

5. Use Project Editor to start Word and create your document.
6. Choose Paragraph from the Format menu, and select the {vflid137438953483}Keep With Next {vflid-3532125269655520} check box. This is the formatting that defines an NSR. Click on OK.
7. Enter the text:  
This text is in the non-scrolling region. It will remain at the top of the master pane as the following text lines are scrolled.
8. Drop down to the next line, and turn off the Keep With Next formatting.

Enter the text:  
This is line 1.

9. Insert additional lines, typing This is line x each time, until you have 30 lines.
10. Save your document file and compile and test your application as usual.
11. Scroll through the topic using the vertical scroll bar. The result should look like Figure 5–18.

```
{ewc vwrt2, TsTextButton, "Figure  
5½18"[Macro=JI('viewerht.mvb>SecWin', `fig5_18')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

Viewer treats any paragraphs that have the Keep With Next paragraph formatting as part of the non-scrolling region. This can include a Viewer picture command—this has a graphic in the non-scrolling region.

The definition of the master pane includes specifying the NSR background color, position, and border separator. The NSR can be located at the top or bottom of the pane.

### Comment

The background color of the NSR can be changed by executing the Viewer {vfld137438953483}MasterNSRColor{vfld12232066859008} command. This would normally be included in topic entry commands. The NSR position and border can similarly be changed by executing the Viewer {vfld137438953483}MasterAspect{vfld8142789060096688128} command.

All NSR paragraphs must be located at the start of the topic, regardless of the NSR position selected. There cannot be any scrolling paragraphs between the NSR paragraphs.

The NSR is commonly used for displaying the topic title, table column headings, or a graphic button bar with multiple hot spots.

**Control {vfld2305852355062530058}Word Wrap{vfld-9079245303906828288}?**

Complexity: EASY

**Problem**

I want to display some text that is wider than the pane. I don't want the words to wrap around to following lines.

**Technique**

Non-wrapping text is defined by applying a special paragraph format option.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to Sections 3.1 and 3.2.)

**Steps**

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP5\_8, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP5\_8. Create these four directories for all projects, even if some are not needed.
2. Use Project Editor to create a new project file in your VIEWERHT\CHAP5\_8 directory. Enter the name of your document file as TEXT\CHAP5\_8.RTF.

Next create the document file:

3. Use Project Editor to start Word and create your document.
4. Choose Paragraph from the Format menu and select the {vfld137438953483}Keep Lines Together{vfld1549382866184437760} check box. Click on OK to return to the document.
5. Copy the text of steps 1 to 4 above into the document. Press [ENTER] as needed to keep the lines from being excessively long.
6. Save your document file and compile and test your application as usual.
7. Reduce the size of the Viewer window until the horizontal scroll bar appears. The window should look like Figure 5-19.

```
{ewc vwrht2, TsTextButton, "Figure
5½19"[Macro=JI('viewerht.mvb>SecWin', `fig5_19')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

**How It Works**

Viewer treats any paragraphs that have the Keep Lines Together paragraph formatting as non-wrapping text. This can include a Viewer picture command—this has a graphic that can be scrolled horizontally.

Wrapping and non-wrapping paragraphs can be mixed within a topic if desired. Viewer displays a horizontal {vfld137438953482}scroll bar{vfld3940108508069888} only if there is material to be displayed that is



wider than the master pane.

**Comment**

Non–wrapping text cannot be used within regular panes. Any text that won't fit within the pane is dropped.

This feature is commonly used to display tables or similar material that must retain its relative positions to be meaningful. It can also be used for topics that contain long lines of definitions—the desired line can be located by the left portion of the material, and the rest can be read by scrolling. This can be especially useful for technical material such as computer programming that may be copied into another document.

Tables are never wrapped, regardless of formatting.

## Use {vflid2305852355062530058}Tables{vflid-9079245303906828288} to Show Information?

Complexity: INTERMEDIATE

### Problem

I have some information that I want to display in a table. How can I make this look good?

### Technique

You use the standard Word support for tables; there are some formatting options that Viewer does not support, as well as some limited workarounds.

You create the standard directories, project file and document file for this How-To. (To review those procedures, refer to Sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP5\_9, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP5\_9. Create these four directories for all projects, even if some are not needed.
2. Use Project Editor to create a new project file in your VIEWERHT\CHAP5\_9 directory. Enter the name of your document file as TEXT\CHAP5\_9.RTF.

Next create the document file:

3. Use Project Editor to start Word and create your document.
4. Choose Insert Table from the Table menu and create a table with two columns and two rows. Leave the Column Width at Automatic.
5. Insert text in each cell, using appropriate forms of the following:  
This text is in the cell in the first row, first column.
6. Insert a few blank lines, then copy the entire table created in steps 4 and 5 at the new insertion point.
7. Select the entire first table, then choose Border from the Format menu. Select Preset Grid and click on OK.
8. Select the first cell of the second table, then choose Border from the Format menu. Select Preset Box and click on OK. Repeat this for each cell in the table.
9. Insert a few blank lines, then choose Paragraph from the Format menu. Set Indentation to 2 inches from the left, and -2 inches for the first line. Click on the Tabs button and change the first tab stop to 2 inches. Click on OK to return to Word.
10. Enter the following text:  
Row 1[TAB]This text will wrap as necessary if you reduce the size of the window.[ENTER]  
Row 2[TAB]This text will also wrap as required. Give it a try! Shrink

the window and see what happens.[ENTER]

11. Save your document file, and compile and test your application as usual. Ignore warning message 4652 from the compiler. See what borders appear in the first two tables.
12. Reduce the size of the Viewer window and see what each table looks like. The window should look like Figure 5–20.

```
{ewc vwrht2, TsTextButton, "Figure  
5i½20"[Macro=JI('viewerht.mvb>SecWin', `fig5_20')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

The first table generates a compiler message:

```
{vfld2305858952132296716} Warning 4652{vfld-9079242005371944960}:  
Table formatting too complex. This is because Viewer does not support grid  
borders in a table.
```

No {vfld137438953484} borders {vfld-35322352208183296} appear in the second table either, although there aren't any messages. Viewer doesn't support borders around table cells, either!

The text in the tables doesn't wrap as you reduce the size of the window. A horizontal scroll bar appears when needed, instead.

The text in the last section does wrap as the window size is reduced. This technique works best for two-column tables, because only the text in the last "column" wraps.

### Comment

There is no way to create borders around table cells in Viewer. Borders can only be created around the outside of the table. You should leave enough space between columns so that the entries remain distinct.

**Create a Welcome... Topic?**

Complexity: INTERMEDIATE

**Problem**

I want to create a special screen that appears when my application is loaded to welcome the user and provide some special instructions.

**Technique**

You demonstrate two different methods for performing this task. You explain when each would be appropriate in the following Comment section.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to Sections 3.1 and 3.2.)

**Steps**

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP5\_10, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP5\_10. Create these four directories for all projects, even if some are not needed.
2. Use Project Editor to create a new project file in your VIEWERHT\CHAP5\_10 directory. Enter the name of your document file as TEXT\CHAP5\_10.RTF.

Next use a Viewer command in the project file to display the topic:

3. Choose Title Options from the Section menu.
4. Enter contents in the Contents Topic field and click on OK.
5. Choose Groups from the Section menu.
6. Click on the New button and clear the Searchable check box. Click on OK.
7. Choose Config from the Section menu.
8. Place the insertion point after the existing script lines and click on the Paste Command button. Select the `{vfld137438953483}JumpID{vfld1477325272146509824}` command and click on OK.
9. Click on the Edit Command button, and set the fields as follows: Enter `qchPath` as the TitleFile, without quotes. Enter one space as the Window Name, without quotes, and enter ``first'` as the Context. Click on OK.
10. Your configuration script should look like Figure 5–21.

```
{ewc vwrht2, TsTextButton, "Figure
5i½21"[Macro=JI('viewerht.mvb>SecWin', `fig5_21')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

11. Save your project file.
12. Use Project Editor to start Word and create your document. Use Topic

Editor to create a new topic with context string contents. Be sure to delete the page break that is created before your topic.

13. Use Topic Editor to add this topic to Browse group Group1, sequence 010.

14. Enter text:

This is the Contents topic.

15. Create a new topic with context string first. Enter text:

WELCOME TO VIEWER!

16. Create a new topic with context string second. Add this topic to Browse group Group1, sequence 020. Enter the text below.

This is the second topic.

17. Your document should look like Figure 5–22. Save your document file, and compile and test your application as usual.

```
{ewc vwrht2, TsTextButton, "Figure  
5½22"[Macro=JI('viewerht.mvb>SecWin', `fig5_22')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

18. You should be greeted with the Welcome topic. Click on the Contents button, and you should see the Contents topic. The Browse buttons should then take you between the Contents and Second topics without redisplaying the Welcome topic.

Next create the same effect with a Topic Entry command:

19. Choose Config from the Section menu.

20. Erase the command you entered earlier by selecting that command and pressing [DEL]. Click on OK.

21. Save the project file, then double-click on the name of the document file to start Word.

22. Position the insertion point after the existing footnote codes (#+) at the beginning of the Contents topic. Insert a Topic Entry command here. The command is so complex that it is easier to insert the footnote directly, without the assistance of Topic Editor.

23. Choose {vfld137438953482}Footnote{vfld12232066859008} from the Insert menu. Click on the {vfld137438953483}Custom Footnote{vfld-3532125269655520} Mark check box, and enter an exclamation point (!) in the field following the check box. Click on OK.

24. Word opens the Footnote window at the bottom of your screen, with the insertion point positioned immediately after an exclamation point. Enter the following text on a single line:

```
IfThen(Not(IsMark("first time")), "SaveMark(`first time');JumpID(qchPath, `first')"  
)
```

25. Be certain that you enter that text exactly as shown. Your document

should look like Figure 5–23.

```
{ewc vwrht2, TsTextButton, "Figure  
5½23"[Macro=JI('viewerht.mvb>SecWin', `fig5_23')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

26. Click on the Close button at the top of the Footnote window.
27. Save your document file, and compile and test your application as usual. It should operate exactly as the first version did.

### How It Works

The first version, using the command in the Config section, has the simplest operation. When Viewer first loads an application, it executes the commands listed in the Config section script. It then defaults to displaying the Contents topic if one is specified. If none is specified, it defaults to displaying the first topic in the file. You inserted a command at the end of that script that forces a jump to the topic with context string *first*. This overrides the default actions. You did not provide any other way to display this topic, so once you left it you did not have any visible way to return. You could return by using the History button.

The second version is far more complex. It lets Viewer display the Contents topic, causing the Topic Entry commands you entered with the exclamation point footnote to be executed. The operation of this multilevel command proceeds as follows:

1. The `{vfld137438953483}IfThen{vfld801785328040935424}()` command has two parts—a condition to be tested, and one or more commands to be executed if the condition is true.
2. The condition—`{vfld137438953483}Not{vfld12232066859008}({vfld137438953483}IsMark{vfld-9223356093936173056}("first time"))`—checks to see if a mark with the name "first time" has been saved. The `Not()` portion means that the commands in the second part execute only if the mark has *not* been saved.
3. If the condition is true (i.e., the mark has not been saved) Viewer executes two commands. The first saves the "first time" mark, and the second forces a jump to the topic with context string *first*.

The first time the Contents topic is displayed, the mark has not been saved and so the commands are executed. Every time the Contents topic is displayed thereafter, the mark has been saved and the commands are bypassed. The user cannot see or change these marks.

### Comment

There is one problem with the simpler first method. All commands in the Config section script are executed when a secondary window is displayed. As a result, using this method in an application that uses secondary windows would cause the main window to redisplay the Welcome topic whenever a secondary window is displayed. This would certainly confuse the user! Use the second method if you have secondary windows, and the first method

otherwise.

The topic entry command in the second method is one of the few you are likely to encounter that is too complex for Topic Editor. This uses commands nested within commands as well as multiple commands separated by a semicolon.

The topic entry command demonstrates a valuable technique for controlling the operation of an application. Marks can be saved and removed as desired, thus prohibiting or permitting other actions. This could, for example, be used in a training application to prevent the user from displaying certain material more than once.

## Start Another Program?

Complexity: EASY

### Problem

I need to start other programs from within Viewer, as a result of actions by the user.

### Technique

You demonstrate starting another program when the user clicks on an author-defined button, clicks on a hot spot, or displays a particular topic.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to Sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP5\_11, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP5\_11. Create these four directories for all projects, even if some are not needed.
2. Use Project Editor to create a new project file in your VIEWERHT\CHAP5\_11 directory. Enter the name of your document file as TEXT\CHAP5\_11.RTF.

Next define the new button:

3. Choose Config from the Section menu. Position the insertion point on the blank line after the existing commands.
4. Click on the Paste command button and select the `{vfld137438953483}CreateButton{vfld1477325272146509824}` command. Click on OK.
5. Click on the Edit Command button, and enter ``btn_run'` as the ButtonID, and ``&Run'` as the ButtonCaption. Be sure to include the left and right single quotes around each entry. Click on the down arrow alongside the Command field and select the ExecProgram command. Edit the entry, replacing the term ``CommandLine'` with `"sol.exe"`, and replacing ProgramState with 0. Include the double quotes around the program name instead of the single quotes inserted by Topic Editor. Click on OK.
6. Choose Groups from the Section menu. Click on the New button to define Group1, and clear the Searchable check box. Click on OK.
7. Save your project file.

Next create the document file and a hot spot:

8. Use Project Editor to start Word and create your document
9. Use Topic Editor to add this topic to Browse Sequence group Group1 with sequence number 010.



10. Drop down a line and enter the text:  
Start Solitaire

11. Select the text you just entered, call up Topic Editor, and select Hot Spot (text). Select the Jump to element.
12. Click on Hidden Text is Command.
13. Click on the Paste Command button, select the {vfld137438953483}ExecProgram{vfld1477325272146509824} command, and click on OK.
14. Click on the Edit Command button, and enter `sol.exe' as the CommandLine, and 0 as the ProgramState. (You can click on the down arrow and select the ProgramState value from the drop-down list if desired.) Click on OK.

Next create a topic that runs a program every time it's displayed:

15. Insert a new topic with context string second.
16. Use Topic Editor to add this topic to Browse Sequence group Group1 with sequence number 020.
17. Use Topic Editor to insert a Topic Entry command. Paste in the ExecProgram command, edit it as in step 14, and click on OK. Enter the text:

Solitaire just started!

18. Save your document file, and compile and test your application as usual. You should be able to start the Solitaire game by clicking on the new Run button, by clicking on the hot spot, or by using the Browse buttons to switch to the second topic. (Close each version of Solitaire as it starts to keep your system from becoming cluttered.) Your test should look like Figure 5-24.

```
{ewc vwrht2, TsTextButton, "Figure  
5½24"[Macro=JI('viewerht.mvb>SecWin', `fig5_24')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

The Viewer {vfld137438953483}ExecProgram{vfld973058453322858496} command executes any Windows or DOS program. The CommandLine parameter follows the same rules as the Program Manager's File, Run menu entry: you can list an executable file, a PIF file, or a data file with an extension that has been associated with an executable program. The ProgramState parameter causes the program to be started in normal, maximized, or minimized mode.

The {vfld137438953483}CreateButton{vfld3940108508069888} command uses three parameters—an internal name for the button, which can be used to remove or redefine the button; a caption for the button; and a

command to be executed when the button is selected.

**Comment**

It is possible to coordinate the actions of a Viewer application and a separate program, but this requires significant programming efforts.

As you saw, the definitions for each method of starting a program—through a button, a hot spot, or a Topic Entry command—are extremely similar. The same concepts apply to executing any other commands through the same methods.

## 5.12 Tips and Tricks

- ⇒ You can create a formatted table in Excel, and insert it in your Word file as an object. Viewer treats this as a pasted picture that retains all the original appearance.
- ⇒ Superscript, subscript, and other unsupported character formats can be included by creating bitmap images of the desired text or by inserting objects. Microsoft's Equation Editor is useful for creating superscript and subscript text.
- ⇒ Although Word only lists fonts supported by your printer, you can type in the name of any other font supported by Viewer. For example, the System font is used in the electronic book on the enclosed CD-ROM disk.
- ⇒ Viewer does not support Word's default tabs; you must define tabs through Word's menu. This is also needed if you use hanging indents, such as in bulleted lists.
- ⇒ Word places a border around consecutive paragraphs with common formats; Viewer places borders around each paragraph individually. Use soft carriage returns to get the same effect.
- ⇒ The techniques used throughout this chapter can serve many other purposes. For example, they form much of the basis for developing a custom author-designed user interface.
- ⇒ The conditional commands used in How-To 5.10 can provide a powerful capability for controlling a Viewer application. Besides Viewer commands such as IsMark, you can also execute any external commands or Windows API functions that return a True/False value. Any such external routines must be defined with a RegisterRoutine command.
- ⇒ Examine any Viewer applications you can try out, and try to determine how the author created various effects. You will find that some of the most exciting applications use techniques than can be performed with little or no programming. The tools demonstrated here can be used in many ways. They are limited only by your imagination.



#.This is Times New Roman.¶

**This is Times New Roman Bold.**¶

This is Arial.¶

**This is Arial Bold.**¶

*This is Script.*¶

***This is Script Bold.***¶

This is Symbol: αβχδε¶

This is WingDings: ☺☹ⓂⓃ¶

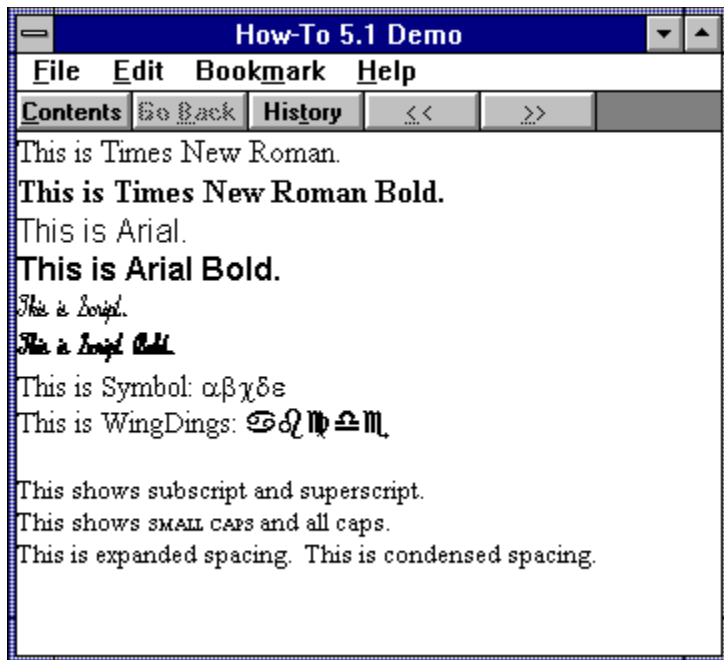
¶

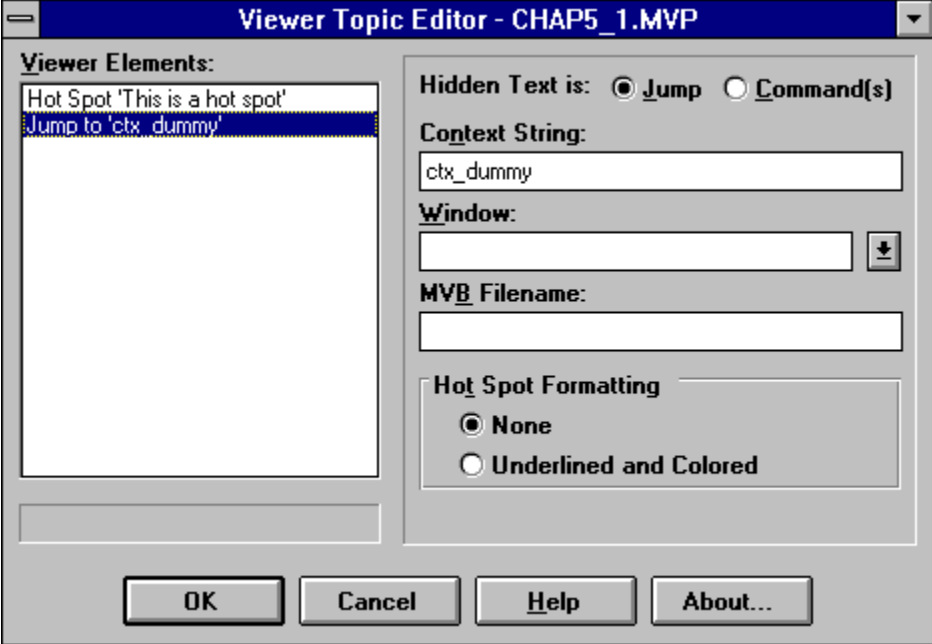
This shows subscript and superscript.¶

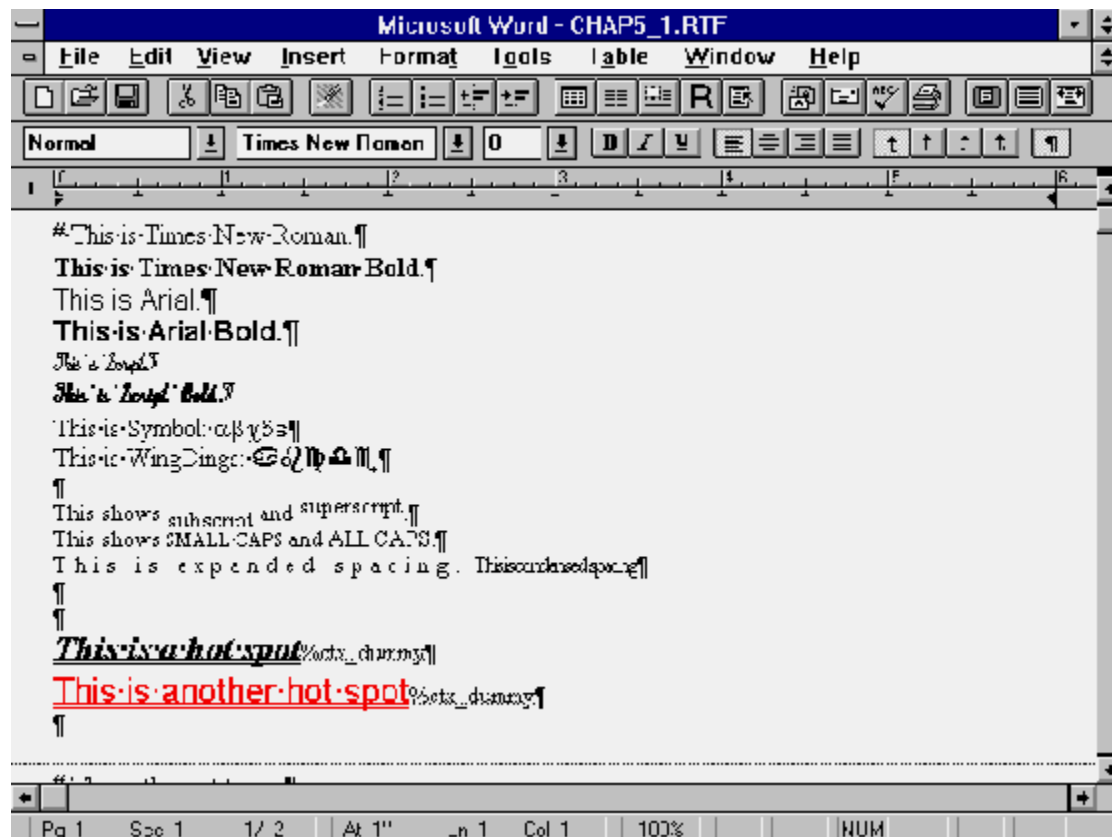
This shows SMALL CAPS and ALL CAPS.¶

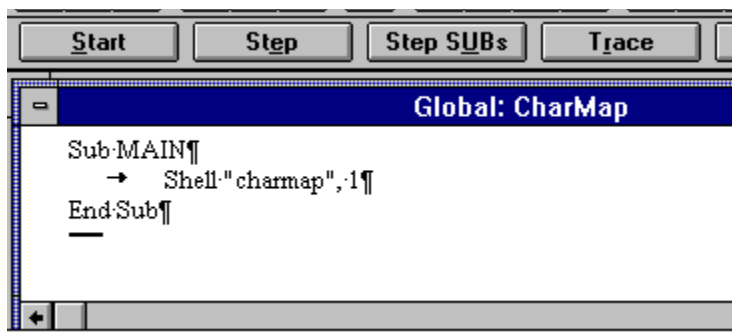
This is expanded spacing. Thiscondensedpace¶

¶

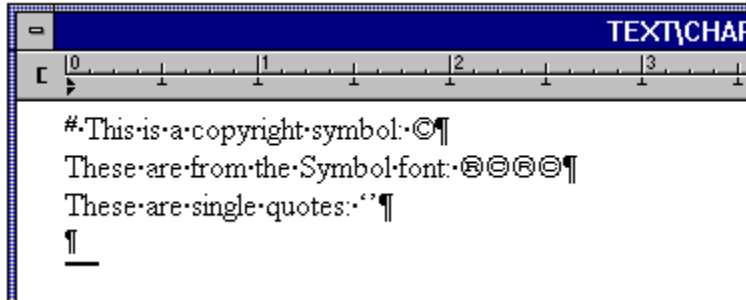


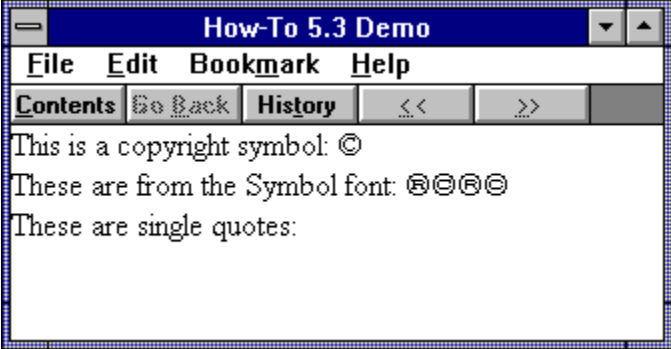




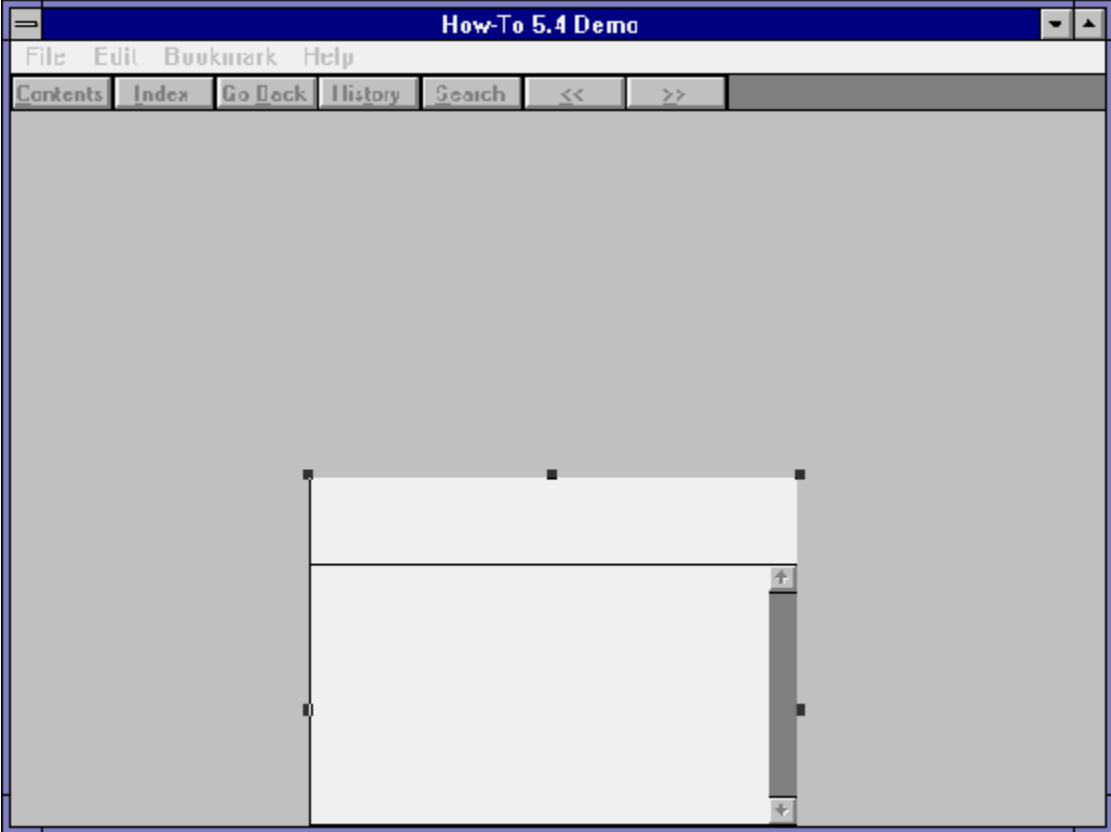


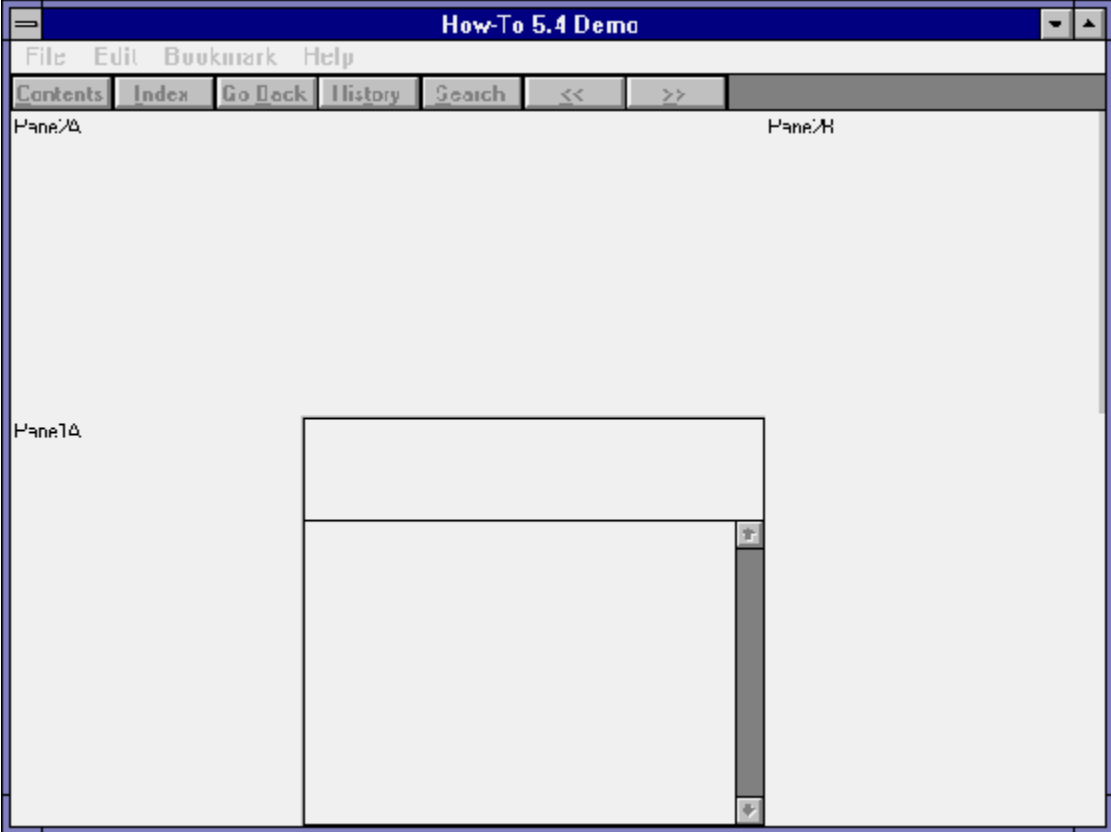


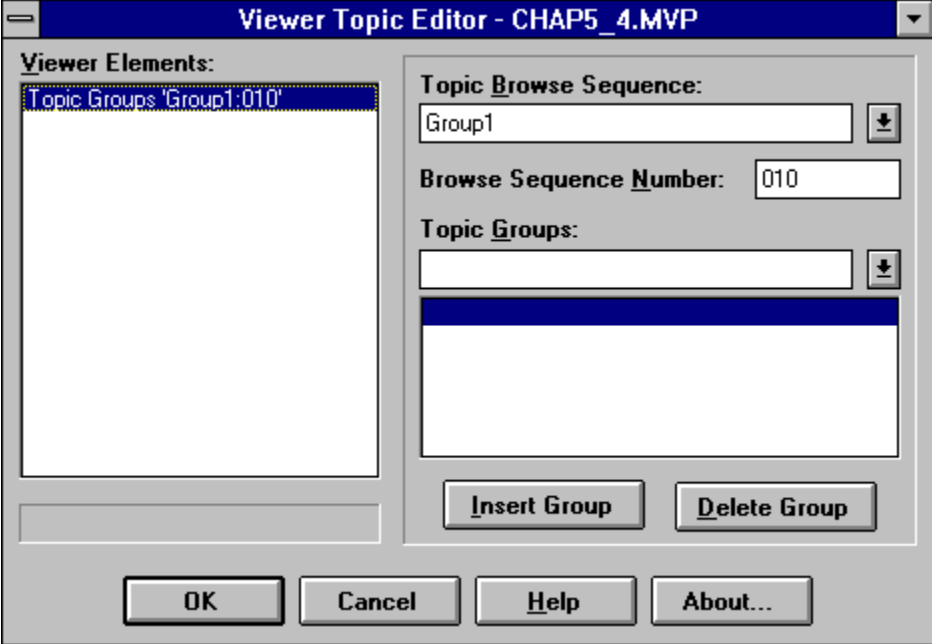




```
Notepad - CHAP5_3.RTF
File Edit Search Help
{\rtf1\ansi \deff0\deflang1024{\fonttbl{\f0\froman Times New Roman;}{\f1\
\red255\green0\blue255;\red255\green0\blue0;\red255\green255\blue0;\red2!
\red192\green192\blue192;}{\stylesheet{\fs20\lang1033 \snext0 Normal;}}{\
\paperw12240\paperh15840\margl1800\marginr1800\margt1440\marginb1440\gutter0
'a9
\par }{\plain \lang1033 These are from the Symbol font: }{\plain \f1\lang
\par }{\plain \lang1033 These are single quotes: }{\plain \lang1033 \lqu
\par }
\par }
```







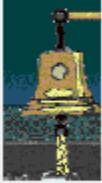
How-To 5.4 Demo

File Edit Bookmark Help

Contents Go Back History << >>

This is part of the first topic. It is displayed in regular pane PanelA.

This is the first topic. It is displayed in the main pane




**This picture goes with the first topic. It is displayed in regular**

How-To 5.4 Demo

File Edit Bookmark Help

Contents Go Back History << >>

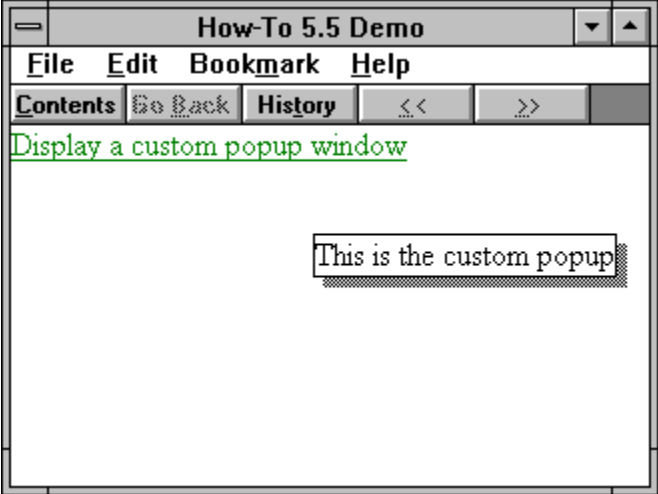
This is part of the second topic. It is displayed in regular pane Pane2A.



This picture goes with the second topic. It is displayed in regular pane Pane2D.

This is the second topic. It is displayed in the main pane.





**Window Properties**

**Window Name:**

**Top:**       **Height:**

**Left:**       **Width:**

**Background Color:**  ...  **Use Default Color**

**Background Picture:**  ...

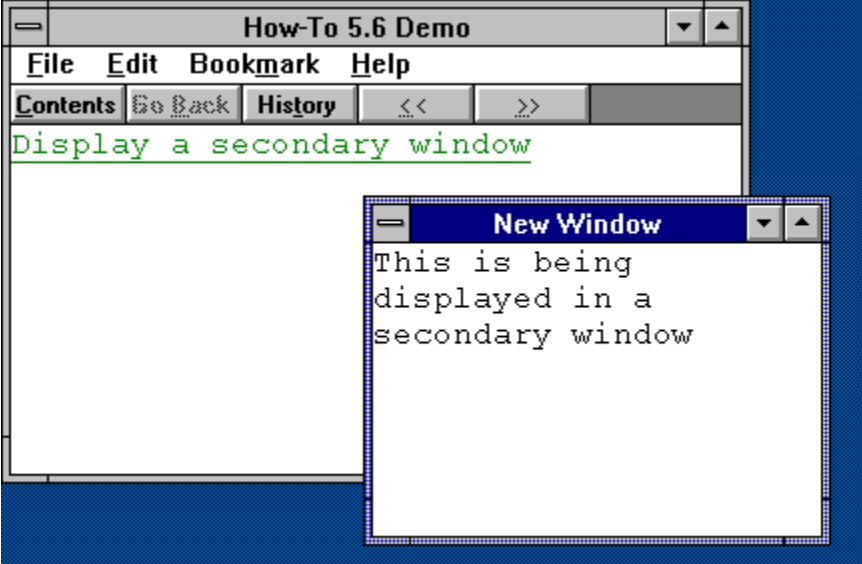
**Window Caption:**

**Initial State:**  ▾

**Stay On Top:**  ▾

**Minimize with MAIN:**  ▾

**Coordinate System:**  ...



**Master Pane Properties**

Pane Name: Master Pane

Top: 457 Max Height: 575

Left: 263 Max Width: 458

Auto-Position

Background Color:   Use Default Color

Border: One-Pixel

**Non-Scrolling Region**

Background Color:   Use Default Color

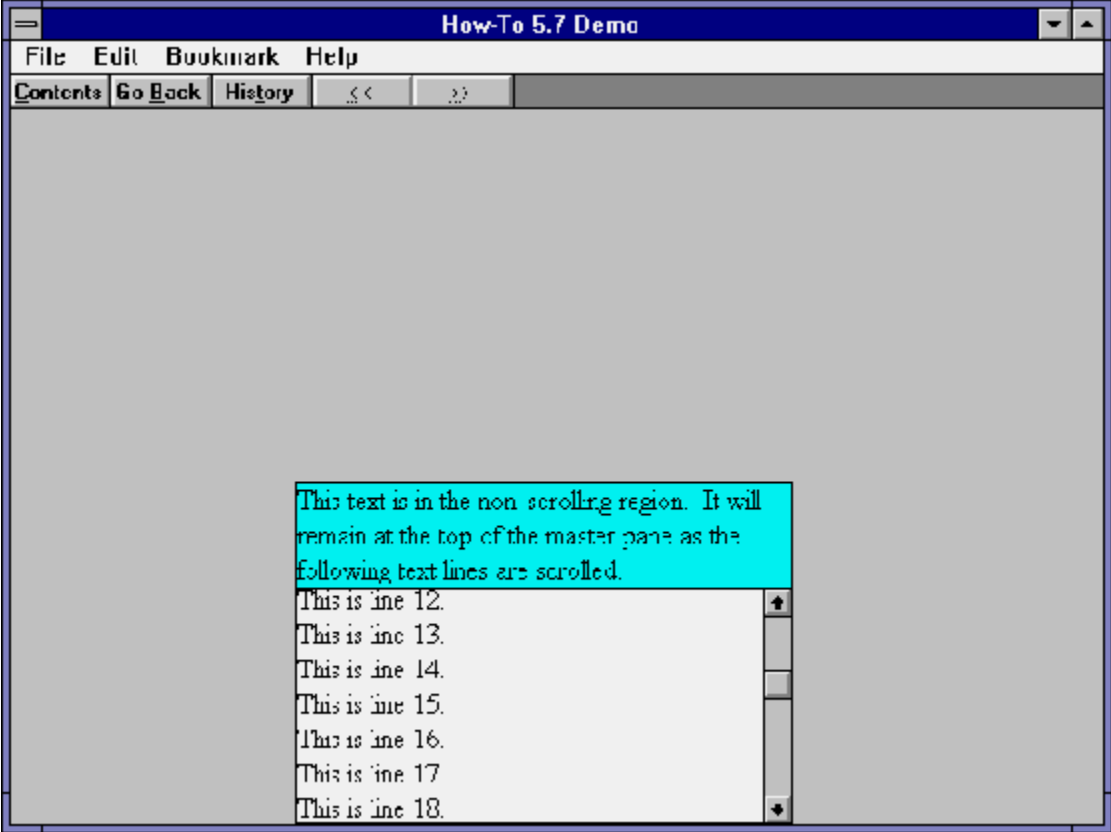
Position: Top

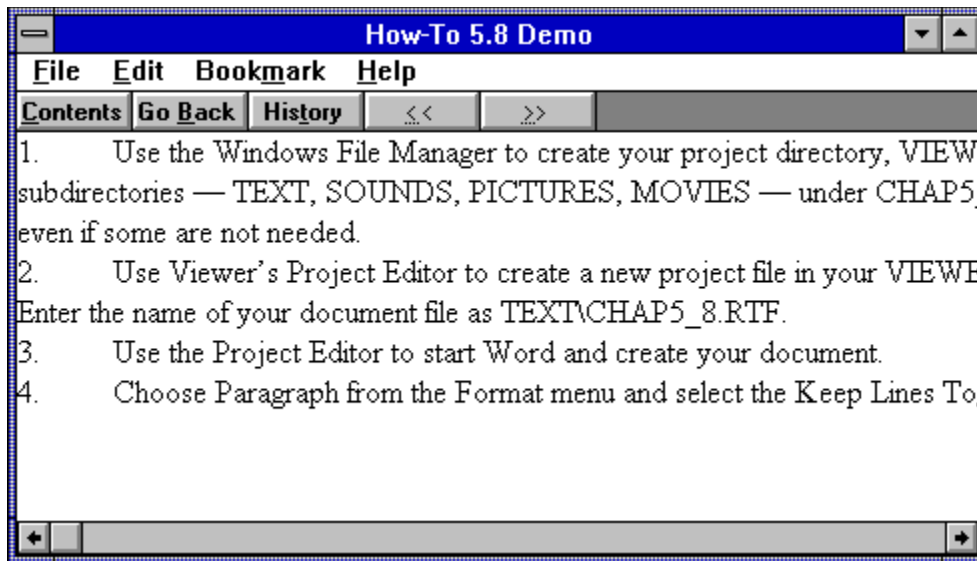
Separator: One-Pixel

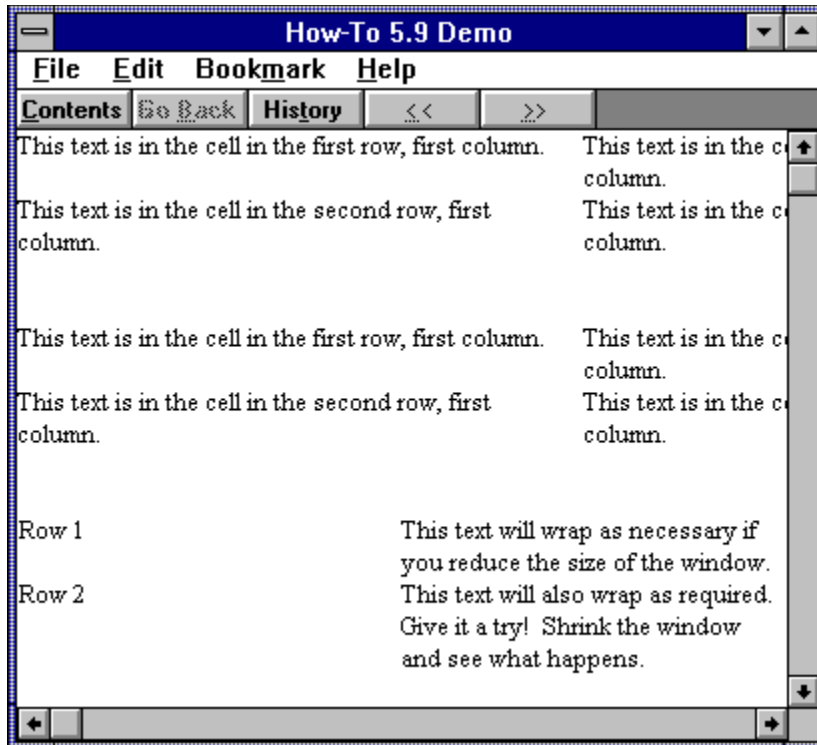
**Minimum Margin**

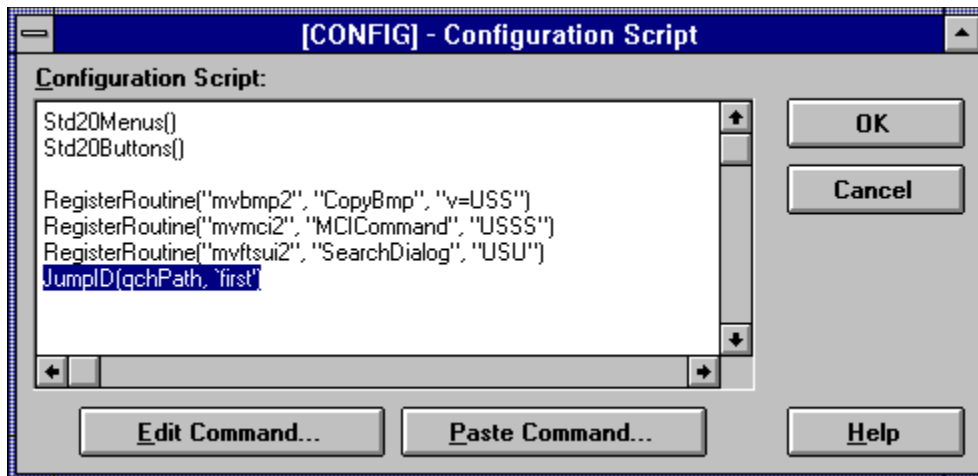
Horizontal: 0 Vertical: 0

Coordinate System: Device-Independent (1024 x 1024)

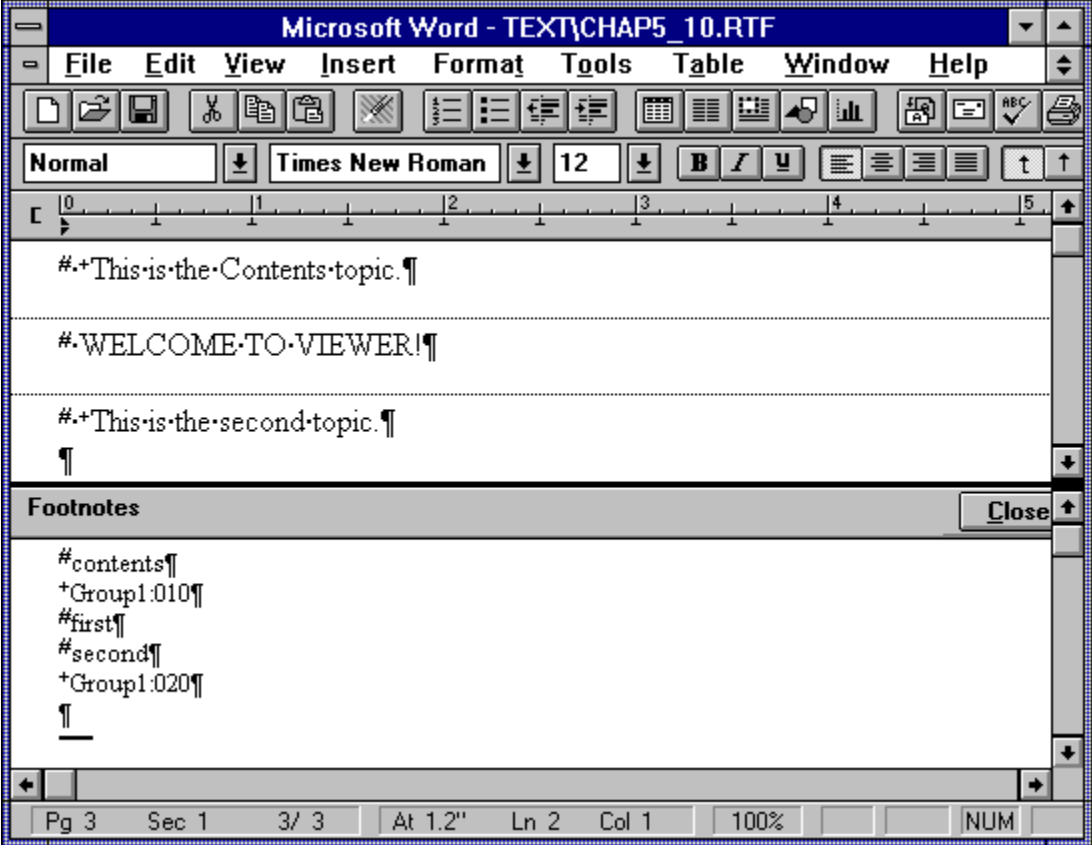


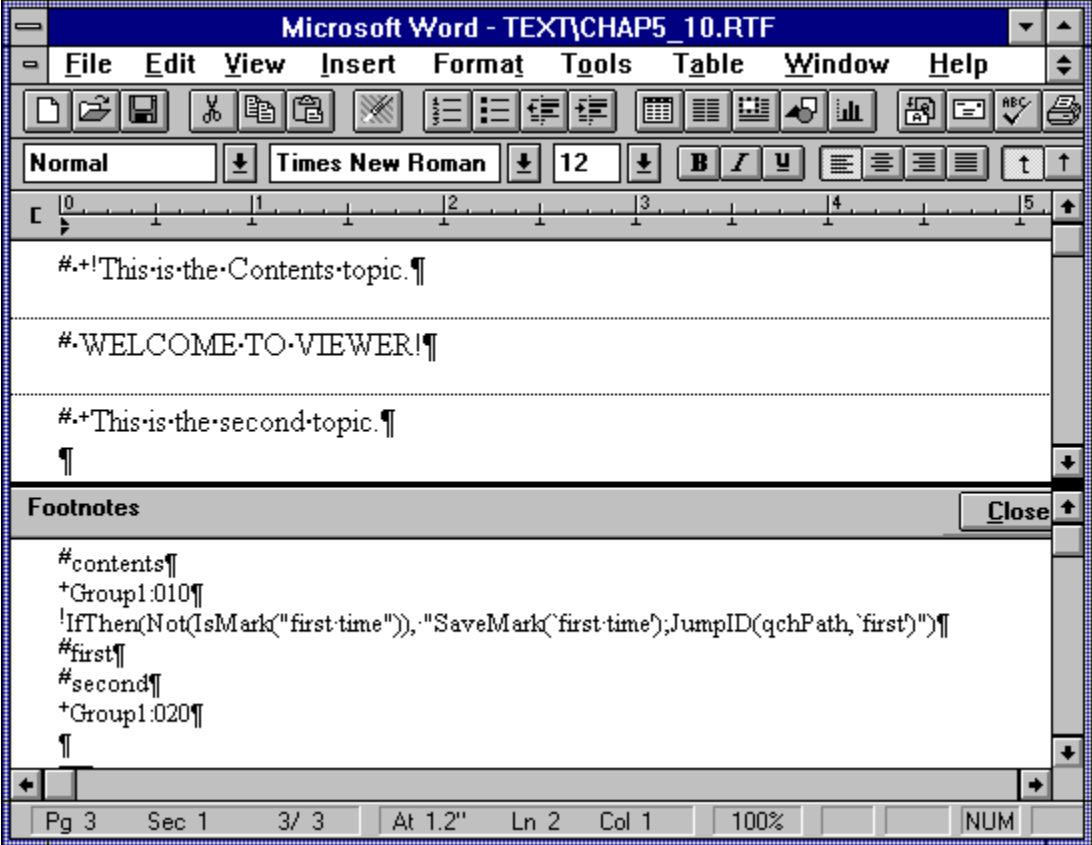


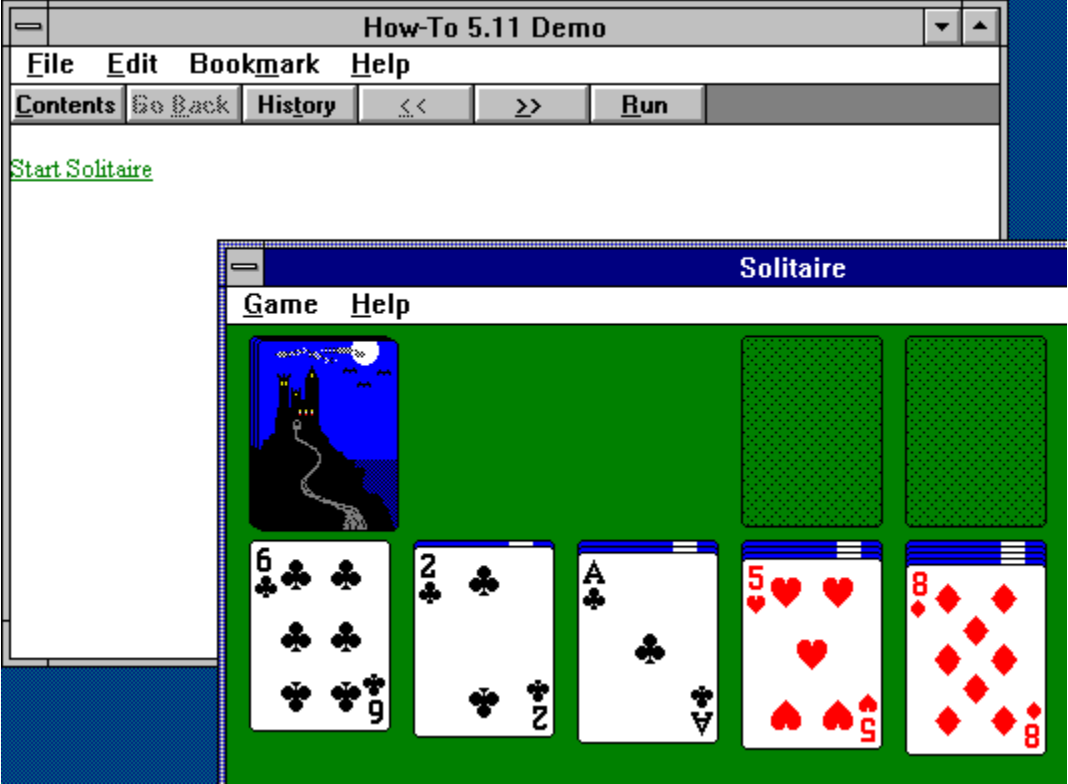














One of the most important parts of a Viewer application is the user interface—the tools you provide for the user to interact with the application. Viewer lets you control much of that interface very easily. The major parts of the interface that you can control are

- ü The button bar. Viewer provides a standard set of buttons. You can include all, some, or none of the standard buttons. You can also add your own.
- ü The menu bar. Viewer provides a standard set of menu titles and items. You can include all, some, or none of the standard menu titles and items. You can also add your own menu titles and items.
- ü Accelerator keys. You can define combinations of keys to execute any Viewer command.
- ü Graphic controls. You can use your own pictures in place of or in addition to the standard menu and button interface.
- ü Adaptable controls. You can replace, change, remove, enable, and disable standard and author-defined buttons, menus, and picture controls while the application is viewed, to reflect changes in the options available based on the user's actions or the portion of the application currently displayed.

Viewer gives you great flexibility, limited only by your imagination. Any Viewer command can be executed using buttons, menus, picture controls, or accelerator keys. You can also use text or graphic hot spots within the topics to execute these commands. Graphic controls can be positioned in several different ways—in the non-scrolling region of the master pane, in the body of the master pane, in regular panes, in secondary windows, or even in popup windows.

This chapter demonstrates the techniques of working with buttons, menus, accelerator keys, and graphic controls. These techniques can be combined with each other and with text and graphic hot spots.

When designing your interface, remember to maintain a simple, clear, consistent set of operations. The user should never have to hunt around for the controls, or wonder what a particular control does.

This chapter uses techniques introduced in earlier chapters, and requires a thorough familiarity with the routine use of Project Editor and Topic Editor. The common operations are described in less detail than in previous chapters, to place greater emphasis on the new material.

**Create an Up {vflid137438953482}Button{vflid3800470531342336}?**

Complexity: INTERMEDIATE

**Problem**

My application includes several logical levels of information—major topics containing subtopics that contain minor topics. I need to create a button that brings the user to the next higher logical level from any topic.

**Technique**

You create a button that causes the appropriate higher-level topic to be displayed. This button is redefined as each topic is displayed so that it jumps to the proper topic. All standard menus and buttons are retained.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

**Steps**

First prepare the directories and project file:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP6\_1, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP6\_1. Create these four directories for all projects, even if some are not needed.
2. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP6\_1 directory. Enter the name of your document file as TEXT\CHAP6\_1.RTF.
3. Choose Title Options from the Section menu to display the Title Options dialog box. Enter contents in the Contents Topic field. Click on OK.

Next create the button:

4. Choose Config from the Section menu, and place the insertion point on the blank line below the existing commands.
5. Click on the Paste Command button, select the {vflid137438953483}CreateButton{vflid1477325272146509824} command, and click on OK.
6. Click on the Edit Command button, and enter `btn\_up' in the ButtonID field, and `&Up' in the Button Caption field. Be sure to include the proper left and right single quotation marks (') around each entry. Click on the down arrow alongside the Command field, then select the Contents() command. The dialog box should look like Figure 6-1.

```
{ewc vwrht2, TsTextButton, "Figure
6i;½1 "[Macro=JI('viewerht.mvb>SecWin', `fig6_1')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

7. Click on OK to return to the Config dialog box, which should look like Figure 6-2. Click on OK to return to the main Project Editor window.

{ewc vwrht2, TsTextButton, "Figure  
6i;½2"[Macro=JI('viewerht.mvb>SecWin', `fig6\_2')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}

Now create the document file:

8. Use Project Editor to start Word and create your document, and create a new topic with the context string contents. Be sure to delete the page break that is created before your topic.
9. Use Topic Editor to define a Topic Entry command. Click on the Paste Command button and select the {vfld137438953483} DisableButton {vfld-9223349496866406400} command. Click on the Edit Command button, and enter `btn\_up' in the ButtonID field. Be sure to include the proper left and right single quotation marks. Click on OK twice to return to the document.
10. Insert a few blank lines, and create hot spots using the information in Table 6–1. Indent the text to maintain the appearance shown in the table, using tabs or spaces.

**Table 6–1. Hot spots for How–To 6–1.**

| Hot Spot Text                        | Context String |
|--------------------------------------|----------------|
| Designing a Viewer Application       | chap_2         |
| How Should I Design the Application? | sect_2_1       |
| How to Present Information           | sect_2_1_a     |
| Select Types of Pictures             | sect_2_1_b     |
| Viewer's Files                       | sect_2_2       |
| Required, Created by Author          | sect_2_2_a     |
| Other Files                          | sect_2_2_b     |
| Creating A Simple Application        | chap_3         |
| Start a New Project                  | sect_3_1       |
| Create directories                   | sect_3_1_a     |
| Create Project File                  | sect_3_1_b     |
| Create a Contents Topic              | sect_3_2       |
| Create the Document File             | sect_3_2_a     |
| Create the Footnotes                 | sect 3 2 b     |

11. Use Topic Editor to create a new topic with context string **chap\_2**.
12. Create another Topic Entry command, and paste in the {vfld137438953483} ChangeButtonBinding {vfld12232066859008} (or {vfld137438953483} CBB {vfld-9223349496866406400}) command. You can use either entry—the full or abbreviated names work the same. Edit the command to set the ButtonID to `btn\_up' and the command to `Contents()'. Click on OK. Move the insertion point down one line and paste in the {vfld137438953483} EnableButton {vfld-9223349496866406400} (or EB) command. Edit the command to set the ButtonID to `btn\_up'. Click on OK twice.

13. Enter the topic heading as follows:  
Chapter 2: How is a Viewer Application Created?

14. Insert a few blank lines and copy the portion of the hot spots in the contents that apply to Chapter 2 into this topic.
15. Create a new topic with context string `sect_2_1`. Create CBB and EB Topic Entry commands similar to those in the previous topic. The CBB command field should be `\JumpID(qchPath, "chap_2")`. Note that the inner pair of quotations must be double quotation marks.

16. Enter the topic heading as follows:  
How Should I Design the Application?

17. Insert a few blank lines and copy the two hot spots for this section from the previous topic.
18. Create a new topic with context string `sect_2_1_a` and CBB and EB Topic Entry commands. The CBB Command field should be `\JumpID(qchPath, "sect_2_1")`. Enter the topic heading:

How to Present Information

19. Insert a few blank lines and enter any desired text. Repeat step 18 for section `2_1_a`. Use the identical CBB command because these topics have the same next higher topic.
20. Repeat steps 15 through 18 for section `2_2` with corresponding CBB commands and field entries.
21. Repeat steps 11 through 19 for Chapter 3 with corresponding CBB commands and field entries.
22. Your document should look like Figure 6–3.

```
{ewc vwrht2, TsTextButton, "Figure  
6½3"[Macro=JI('viewerht.mvb>SecWin', 'fig6_3')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

23. Save your document file, and compile and test as usual. Display various levels of the application by using the contents lists, and see what topic is displayed by the new button in each case. Note that the button is disabled at the highest level.

### How It Works

The new button is created when the application is loaded by a command in the Config section of the project file. This command gives the button an internal name (`btn_up`) that can be used by other commands to modify the button, a caption (`&Up`) that allows the button to be selected through the keyboard by pressing `[ALT]-[U]`, and a command to be executed when the button is selected (`Contents()`).

When the Contents topic is displayed, the topic entry command disables the new button, because there is no higher level to display.



When any other topic is displayed, the topic entry commands perform two tasks. The command associated with the button is replaced with a command that displays the proper topic, and the button is enabled (in case you jump to this topic straight from the Contents). The button always displays the higher topic. In the main chapter topics (such as chap\_3) you use the Contents() command. In the major section topics (such as sect\_3\_1) you use a JumpID command with the chapter topic's context string. In the minor section topics (such as sect\_3\_1\_a) you use a JumpID command with the major section's context string.

### **Comment**

This technique of redefining the button's operation in every topic is the standard way to handle any part of the user interface that changes based on what part of the application is displayed. You must redefine the operations in every topic unless you limit the user's ability to jump between topics. Limiting the user requires removing the History, Back, Index, and Search buttons and studying the possible paths through hot spots very carefully. If you miss any possibilities, you guarantee that your users are surprised by the unexpected topics that appear when they use this button.

The only alternative to continuously redefining the buttons is to include the controls as pictures in a non-scrolling region within every topic, with suitable definitions in each case. There isn't much difference—you need a set of controls for each topic either way. If the controls are included within the topics, they display automatically.

## Create a Configuration Menu?

Complexity: DIFFICULT

### Problem

I want to let the user control some operations of the application, such as whether or not to play sounds and to display VGA or SVGA versions of pictures. The user should only be able to change these options while in the Contents topic.

### Technique

You create a new menu title with three items—one to control sounds, one to select VGA pictures, and one to select SVGA pictures. The commands executed by these menu items set checkmarks on the menu to show the active options and set marks that can be tested throughout the application. Marks are place markers that cannot be seen by the user, but can be tested and used by Viewer commands. You disable the menu items outside the Contents topic, and enable them within that topic.

This section uses complex conditional (`{IfThen}`) commands. If you are not familiar with these commands, you should review How-To 5.10.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP6\_2, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP6\_2. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the pictures files for this How-To from the VIEWERHT\HOWTOS\CHAP6\CHAP6\_2\PICTURES directory on the CD-ROM to the \VIEWERHT\CHAP6\_2\PICTURES directory on your hard drive. Copy the sound files from the SOUNDS subdirectory similarly.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP6\_2 directory. Enter the name of your document file as TEXT\CHAP6\_2.RTF.

Next create the menu title and items:

4. Choose Config from Project Editor's Section menu to display the Configuration Script dialog box, and move the insertion point to the first blank line under the existing commands.
5. Click on the Paste Command button, select the `{RegisterRoutine}` command, and click on OK. Click on the Edit Command button and enter 'mmsystem.dll' as the DLLName,

`{vflid137438953483}sndPlaySound{vflid-9223357193447800832}' as the FunctionName, and `Su' as the ParameterSpec. Click on OK.

6. Paste in the {vflid137438953483}InsertMenu{vflid-9223349496866406400} command. Edit the command and enter `mnu\_config' as the MenuID, `Con&figuration' as the Menu Caption, and 4 as the Menu Position.
7. Paste in the {vflid137438953483}AppendItem{vflid-9223349496866406400} command. Edit the command and enter `mnu\_config' as the MenuID, `mnu\_sounds' as theNewItemID, and `So&unds' as the ItemCaption, and enter the following text in the Command field (all as one line):

```
IfThenElse({vflid137438953483}IsMark{vflid12232066859008}
("sounds"),"{vflid137438953483}UncheckItem{vflid12232066859008}
){`mnu_sounds'};
{vflid137438953483}DeleteMark{vflid12232066859008}
(`sounds'),"{vflid137438953483}CheckItem{vflid12232066859008}
(`mnu_sounds');
{vflid137438953483}SaveMark{vflid280933810831360}(`sounds')
```

The completed dialog box should look like Figure 6–4.

```
{ewc vwrht2, TsTextButton, "Figure
6i;½4"[Macro=JI('viewerht.mvb>SecWin', `fig6_4')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

8. Paste in another AppendItem command. Enter `mnu\_config' as the MenuID, `mnu\_VGA' as theNewItemID, `&VGA' as the ItemCaption, and enter the following Command (all as one line):

```
IfThen(IsMark("SVGA"),"UncheckItem(`mnu_SVGA');
CheckItem(`mnu_VGA');DeleteMark(`SVGA'); SaveMark(`VGA')
```

9. Paste in another AppendItem command. Enter `mnu\_config' as the MenuID, `mnu\_SVGA' as theNewItemID, `&SVGA' as the ItemCaption, and enter the following Command (all as one line):

```
IfThen(IsMark("VGA"),"UncheckItem(`mnu_VGA');
CheckItem(`mnu_SVGA');DeleteMark(`VGA'); SaveMark(`SVGA')
```

10. Paste in a CheckItem command, and enter `mnu\_SVGA' as the MenuID.
11. Paste in a SaveMark command, and enter `SVGA' as the MarkID. The completed script should look like Figure 6–5.

```
{ewc vwrht2, TsTextButton, "Figure
6i;½5"[Macro=JI('viewerht.mvb>SecWin', `fig6_5')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

12. Save your project file.

Next create the document file:

13. Use Project Editor to start Word and create your document. You don't have to create a topic here—the beginning of the file is automatically a new topic, and this topic doesn't need a context string.
14. Create a hot spot with text `Play sounds`, jumping to `ctx_sounds`.
15. Create a hot spot with text `Show picture`, jumping to `ctx_picture`.

Next create the topic to play sounds if the Configuration menu selection permits:

16. Create a new topic with context string `ctx_sounds`.
17. Choose Footnote from Word's Insert menu. Choose the Custom Footnote Mark radio button, and enter an exclamation point (!) as the custom mark. Click on OK.
18. Word displays the Footnote window at the bottom of the screen. Enter the following command immediately after the exclamation point:  
`IfThen(IsMark(`sounds'),'sndPlaySound(`sounds\chimes.wav',1)')`
19. Click on the Close button at the top of the Footnote window.
20. Enter text:  
This plays chimes if allowed. Click on the Go Back button to return.

Next create the topics to display pictures.

21. Create a new topic with context string `ctx_picture`.
22. Choose Footnote from Word's Insert menu. Choose the Custom Footnote Mark radio button, and enter an exclamation point as the custom mark. Click on OK.
23. Word displays the Footnote window at the bottom of the screen. Enter the following command immediately after the exclamation point:  
`IfThen(IsMark(`VGA'),'JumpID(qchPath, `ctx_VGA')')`
24. Click on the Close button at the top of the Footnote window.
25. Insert a couple of blank lines and insert a Picture (using `ewX...`) command. Specify filename `PICTURES\SVGA.BMP`, and Caption This is a Super VGA picture.
26. Insert a few more blank lines and enter text:  
Click on the Go Back button to return.
27. Create a new topic with context string `ctx_VGA`.
28. Insert a couple of blank lines and insert a Picture (using `ewX...`) command. Specify filename `PICTURES\VGA.BMP`, and Caption This is a VGA picture.
29. Insert a few more blank lines and enter the following text:  
Click on the Go Back button to return.
30. The completed document should look like Figure 6–6. Save your document file, and compile and test as usual.

```
{ewc vwrht2, TsTextButton, "Figure  
6½"[Macro=JI('viewerht.mvb>SecWin', `fig6_6')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

31. See how the menu checkmarks appear and disappear as you select items. Set the new Sounds menu item on, and click on the Play sounds hot spot. Go back and try this again with the option turned off.
32. Test the picture options similarly. Note that clicking on the Go Back button while the VGA picture is displayed doesn't work—the command entered in step 23 is reexecuted and the VGA topic is redisplayed.

### How It Works

In step 5 you add a command in the Config section of the project file that creates the menu title with the internal name of `mnu_config` and a caption of Configuration, located as the fifth title on the menu bar. The position values start with zero for the first title.

The Sounds menu item contains a complex nested command. If there is a mark named “sounds,” this menu item is unchecked and the mark is deleted. If the mark does not exist, it is created and the menu item is checked. The presence of that mark is tested in the `ctx_sounds` topic—the `sndPlaySound` command is only executed if the sounds mark exists. Note that this item defaults to off—no sounds play until the menu item is selected.

The VGA menu item checks for a mark named “SVGA.” If that mark exists, the SVGA menu item is unchecked, the VGA item is checked, the SVGA mark is deleted, and the VGA mark is created. The SVGA menu item has the corresponding commands in reverse. The `ctx_pictures` topic tests for the VGA mark, and jumps to the `ctx_VGA` topic if the mark exists. If not, the SVGA picture is displayed within the current topic. The Config script includes commands to check the SVGA menu item and create the SVGA mark.

### Comment

An `ewX` command, used to display a picture or play sounds or movies, cannot test for marks. The only way to use marks to control `ewX` commands is by controlling which topics are displayed. As a result, all topics that contain `ewX` commands that you want to control must be duplicated in the document, just as the picture topics were duplicated in this How-To. Commands that are executed in hot spots, buttons, menu items, or topic entry commands can include conditional tests.

Note the multiple levels of quotation marks required to enter the conditional commands in the menu item definitions. Each level of quotations must alternate between single and double quotation marks. The entire command must be enclosed in quotation marks because it is a parameter of the `AppendItem` command. Because you used single quotations for that, you had to use double quotations around the commands within the `IfThenElse` command. The parameters within *those* commands then use single quotations

again.

You use the `AppendItem` command to define the menu items instead of the `InsertItem` command because it's slightly simpler. `AppendItem` adds an item at the end of the menu, whereas `InsertItem` requires that you define the specific position.

## Customize the About... Window?

Complexity: EASY

### Problem

I want to create my own About... window, to be displayed by an item under the Help menu title.

### Technique

You can add one line of  
{vfld137438953482}copyright {vfld4573404880428859392} information to the standard About... window, but cannot change the rest of it. You modify the standard menu item to display a topic within your document file instead of the standard window.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and project file, and create the menu item:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP6\_3, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP6\_3. Create these four directories for all projects, even if some are not needed.
2. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP6\_3 directory. Enter the name of your document file as TEXT\CHAP6\_3.RTF.
3. Choose Config from the Section menu to display the Configuration Script dialog box, and position the insertion point on a blank line. Click on the Paste Command button to display the Paste Command dialog box, select the {vfld137438953483}DeleteItem {vfld-9223349496866406400} command, and click on OK. Click on the Edit Command button to display the Edit Command dialog box, and use the pull-down list to select ItemID mnu\_about. Click on OK.
4. Position the insertion point at the next line in the Config script, and paste in the {vfld137438953483}AppendItem {vfld-9223349496866406400} command. Click on the Edit Command button to set the parameters. Click on the down arrow alongside MenuID, and select mnu\_help. Click on the down arrow alongsideNewItemID, and select mnu\_about. Enter '&About Chap6\_3' for the ItemCaption. For the Command field select PopupID from the pull-down list, then replace the TitleFile entry with qchPath (without quotation marks), and replace Context>PopupName with `ctx\_about`. Click on OK twice.

Next create the document file and the new About topic:

5. Use Project Editor to start Word and create your document. Enter the following text: This is the Contents topic. This represents the entire application.

6. Create a new topic with context string `ctx_about`. Enter a few lines of text as follows, with each sentence on a separate line:  
Multimedia Viewer How-To 6.3. Copyright (C) 1993 Stephen Pruitt.  
This How-To prepared by (your name).
7. Insert a few blank lines and type this text:  
[Click anywhere on the main window to return]
8. Save your document file, and compile and test as usual.
9. Choose About from the Help menu in your application. It should look like Figure 6-7.

```
{ewc vwrht2, TsTextButton, "Figure
6½7"[Macro=JI('viewerht.mvb>SecWin', `fig6_7')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

Steps 3 and 4 remove the standard menu item and replace it with a custom one. The new item executes the `PopupID` command when selected—this displays a popup window just as in a hot spot.

### Comment

Viewer does not allow the command in a standard menu item to be changed with the `{vfld137438953483}ChangeItemBinding{vfld-8970326573957775360} command`—this can only be used on author-defined menu items. Therefore you have to delete and replace the item. This also allows you to change the caption.

The popup window does not look as good as the standard About window, and you can't use an OK button to close it. There isn't any command a button could execute to close the window. A normal-appearing About window could be produced by writing a DLL or Visual Basic program that displays the window in the proper location and closes it when the OK button is clicked. You would execute the appropriate command in the menu item to call your DLL or VB program. The location of the window in such programs is usually determined by the location of the mouse pointer. This can cause strange results if the keyboard is used to select the menu item!

The TSTools Viewer extension, included on the CD enclosed with this book, has a function that can be used to create a professional About box. This is described further in Appendix B. This function is used in the VIEWERHT application that is on the same CD disk.



**Add {vfld137438953482}Accelerator Keys{vfld3800470531342336}?**

Complexity: EASY

**Problem**

I want to allow users to perform several different operations through special key combinations. These include displaying a glossary topic in a secondary window and executing a separate Windows program.

**Technique**

You use the Viewer

{vfld137438953483}AddAccelerator{vfld280933810831360} command to define the key combinations and their associated actions.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

**Steps**

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP6\_4, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP6\_4. Create these four directories for all projects, even if some are not needed.
2. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP6\_4 directory. Enter the name of your document file as TEXT\CHAP6\_4.RTF.

Next define a secondary window and the accelerator keys:

3. Choose Window Definitions from the Section menu. Click on the New button, which creates secondary window Win1. Click on the Properties button to display the Window Properties dialog box, and change the Window Caption field to Glossary. Leave all other fields at their default values. Use the System Control box to close this dialog box. Click OK in the Window Definitions dialog box to return to the main Project Editor window.
4. Choose Config from the Section menu. Place the insertion point on the first blank line after the existing script commands, and click on the Paste Command button. Select the AddAccelerator command, and click on OK.
5. Click on the Edit Command button to display the Edit Command dialog box. Set the Key field by selecting the 0x31 — 1 key entry from the pull-down list. Set the Shift field to 2 — Ctrl by using the pull-down list for that field. Set the Command field to JumpID similarly. Edit the JumpID command parameters so that the command is as shown below (Note the placement of single and double quotations):

```
`JumpID("CHAP6_4.MVB>Win1", "glossary")'
```

Click on OK.

6. Repeat steps 5 and 6 to paste in another AddAccelerator command, selecting keys “2 — Ctrl” and “0x32 — 2.” Select the ExecProgram command. Edit the CommandLine to read "sol.exe", and set the

ProgramState to 0. Click on OK. The resulting script should look like Figure 6–8.

```
{ewc vwrht2, TsTextButton, "Figure  
6½8"[Macro=JI('viewerht.mvb>SecWin', `fig6_8')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

7. Use Project Editor to start Word and create your document. Enter a line of text: This is the Contents topic. This topic is used, for this How-To, to represent a complete normal application.
8. Create a new topic with the context string `{vfld2305865549202063370}glossary{vfld-9079242005371944960}`. Enter a series of lines of text—each beginning with a term from this book, followed by a short definition or description. For example, terms from this chapter could include Button bar, Menu title, Menu item, Accelerator key, AddAccelerator, InsertMenu, AppendItem, ExecProgram, ChangeButtonBinding, CheckItem, and UncheckItem. The entries should be listed in alphabetic order.
9. Save your document file, and compile and test as usual.
10. Press **[CTRL]–[1]**—a secondary window containing your glossary should appear. Press **[CTRL]–[2]**—the Windows Solitaire game should start. The results should look like Figure 6–9.

```
{ewc vwrht2, TsTextButton, "Figure  
6½9"[Macro=JI('viewerht.mvb>SecWin', `fig6_9')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

The **[CTRL]–[1]** accelerator key defined in steps 4 and 5 executes a common JumpID command that specifies a secondary window. When that key combination is pressed the JumpID command is executed, creating a secondary window containing the glossary topic.

The **[CTRL]–[2]** accelerator key defined in step 6 executes an ExecProgram command that starts the Solitaire program. This is very similar to the examples in How-To 5.11.

### Comment

Accelerator keys must be easy to remember unless there is a visible list of the keys. They should not include the **[ALT]–letter** shortcuts used to select menus or buttons.

## Create a Graphic Interface in the Master Pane?

Complexity: INTERMEDIATE

### Problem

I want to use pictures to represent the Contents, Previous, Next, and Exit operations. I want to put them inside my master pane.

### Technique

You create a `{vfld2305858952132296714}non-scrolling region{vfld-9079242005371944960}` (NSR) in the master pane. The pictures are placed in a table within the NSR.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP6\_5, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP6\_5. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the picture files for this How-To from directory VIEWERHT\HOWTOS\CHAP6\CHAP6\_5\PICTURES on the CD-ROM to the VIEWERHT\CHAP6\_5\PICTURES subdirectory on your hard disk.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP6\_5 directory. Enter the name of your document file as TEXT\CHAP6\_5.RTF.
4. Choose Title Options from the Section menu, and enter contents in the Contents Topic field. Click on OK.

Next create the NSR and controls:

5. Use Project Editor to start Word and create your document, and create a new topic with the context string contents. Be sure to delete the page break that is created before your topic.
6. Choose Paragraph from Word's Format menu, and check the Keep With Next check box. Click on OK.
7. Choose Insert Table from the Table menu, and create a table with four columns, one row, and a 1.6-inch column width.
8. Choose Column Width from the Table menu, and set Space Between Columns to 0.
9. Place the insertion point in the first table cell, and use Topic Editor to define a Picture (using ewX) command.
10. Click on the Options button, and select the desired file name—PICTURES\CONTENTS.BMP. Check the Store Picture in Baggage check box.
11. Click on the Paste Command button, and select the Contents() command.

This command has no options, so you don't need to edit it. Click on OK twice to return to Word.

- Repeat steps 9 through 11, inserting picture files PREVIOUS, NEXT, and EXIT with corresponding commands Prev(), Next(), and Exit().
- Insert a blank line, then choose Paragraph from Word's Format menu and uncheck the Keep With Next check box.

Now create the topics to use for testing:

- Enter a line of text: This is the Contents topic.
- Enter 30 lines of text reading This is line 1, etc. The document should look like Figure 6–10.

```
{ewc vwrht2, TsTextButton, "Figure  
6½10"[Macro=JI('viewerht.mvb>SecWin', `fig6_10')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

- Insert a few blank lines, and insert a text hot spot with the text Go to another topic and jumping to context string next.
- Create a new topic with context string next. Insert the text This is a separate topic.
- Save your document, and compile and test as usual. The contents topic should look like Figure 6–11. Scroll through the topic text.

```
{ewc vwrht2, TsTextButton, "Figure  
6½11"[Macro=JI('viewerht.mvb>SecWin', `fig6_11')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

- See what the next topic looks like by clicking on the hot spot, then click on the Exit picture to end the program.

### How It Works

You created the {vfld137438953482}button bar{vfld-35184913254711296} using the techniques demonstrated in How-To 4.2—a table with ewc embedded window commands. Each command is set up to execute an appropriate Viewer command if the user clicks on the picture.

The button bar is placed into a {vfld137438953482}non-scrolling region{vfld8070312552128577536} by formatting it as a Keep With Next paragraph. This defines an NSR, causing the button bar to remain in place as you scroll through the text for that topic.

The Previous and Next pictures don't do anything because you didn't define a Browse sequence. You include that in the next How-To.

### Comment

The thin border separating the NSR from the body of the topic is part of the default definition of the NSR. You could have eliminated the border, changed

the NSR background color, or positioned the NSR at the bottom of the master pane through Window Definitions in Project Editor's Section menu.

The wide spacing between the buttons is caused by the table that was used. How-To 4.2 demonstrates the effect of various techniques for creating button bars.

The graphic button bar doesn't exist in the second topic because you didn't define one there. This type of control must be included in *every* topic to maintain a consistent appearance.

In a real application, such a graphic bar has to change appearance to reflect the available commands. Grayed-out versions of the Previous and Next pictures would be used within topics that are not in a Browse sequence, or are at the start or end of the sequence. This is where the table structure is useful, because it helps to assure that the pictures retain the same position between topics, and do not appear to jump as new topics are displayed.

## Create a Graphic Interface in a {vfid137438953482}Regular Pane{vfid3800470531342336}?

Complexity: INTERMEDIATE

### Problem

I liked the results of the last How-To, but now I want to put my picture controls in a regular pane alongside the master pane.

### Technique

You create a regular pane on one side of the master pane, and display the controls in a vertical table in that pane. You define a Browse sequence this time so all the controls work.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP6\_6, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP6\_6. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the picture files for this How-To from directory VIEWERHT\HOWTOS\CHAP6\CHAP6\_6\PICTURES on the CD-ROM to the VIEWERHT\CHAP6\_6\PICTURES subdirectory on your hard disk.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP6\_6 directory. Enter the name of your document file as TEXT\CHAP6\_6.RTF.
4. Choose Title Options from the Section menu, and enter contents in the Contents Topic field.

Next define the regular pane:

5. Choose Window Definitions from the Section menu, and click on the Properties button to display the Window Properties dialog box. Click on the Master Pane button to display the Master Pane Properties dialog box.
6. Clear the Auto-position check box, then click on the Preview On button.
7. Drag the borders of the Master Pane to expose an area on the right. Double-click on the System Control box to close this window. Close the Window Properties dialog box the same way.
8. In the Window Definitions dialog box, click on the Panes file folder tab then click on the New button. Click on the Properties button to display the Pane Properties dialog box.
9. Change the Pane Name field to Controls.
10. Change the Dismiss When field to Title is Closed, and the Border field to (none).
11. Click on the Windows button to display the Pane Associations dialog

box, then check the Show in Window check box.

12. Click on the Preview button to display the window layout, and then drag and stretch the Controls pane to cover the area to the right of the master pane, as shown in Figure 6–12.

```
{ewc vwrht2, TsTextButton, "Figure  
6½12"[Macro=JI('viewerht.mvb>SecWin', `fig6_12')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

13. Use the System Control box to close the Window Properties dialog box, then click on OK in the Window Definitions dialog box.

Next create a topic group and cause the Controls pane to be displayed immediately:

14. Choose Groups from the Section menu to display the Groups dialog box, and click on the New button.
15. Clear the Searchable check box. Leave the name as the default Group1. Click on OK.
16. Choose Config from the Section menu to display the Config dialog box.
17. Position the insertion point on the blank line below the last existing command, and click on the Paste Command button. Select the PaneID command, and click on OK.
18. Click on the Edit Command button. Enter `{vfld2305878743341596682}qchPath{vfld-9223349496866406400}` in the TitleFile field without quotation marks, and enter a single space in the WindowName field without quotation marks. Enter ``ctx_control`` in the Context field, and ``Controls`` in the PaneName field with single quotation marks for both fields. Enter 0 in the PrintTabCopyOrder field, and click on OK twice to return to the main Project Editor window.

Next create some topics for testing:

19. Use Project Editor to start Word and create your document, and create a new topic with context string contents. Be sure to delete the page break that is created before your topic.
20. Use Topic Editor to define a Topic Group footnote for Group1 and Sequence Number 010.
21. Enter the following text:  
This is the Contents topic. Use the Next button to display the next topic.
22. Insert a blank line and create a new topic with context string topic2. Define a Topic Group footnote for Group1 and sequence 020. Enter the following text:  
This is the second topic. Use the Previous or Contents buttons to display the other topic.

Next create the topic containing the controls:

23. Insert a blank line and create a new topic with context string `ctx_control`.
24. Choose Insert Table from the Table menu, and create a table with one column, four rows, and a 2-inch column width.
25. Select all four rows of the table, then choose Row Height from the Table menu, and set the height to Exactly 12 lines.
26. Position the insertion point in the first table cell, and use Topic Editor to insert a Picture (Using ewX) command. Select the PICTURES\CONTENTS.BMP file, and check the Store Picture in Baggage check box.
27. Click on the Paste Command button, and select the `{vfld137438953483}Contents(){vfld801785328040935424}` command. This command doesn't use any parameters, so you don't need to edit it. Click on OK twice to return to the document.
28. Repeat steps 26 and 27, inserting picture files PREVIOUS, NEXT, and EXIT with corresponding commands `{vfld137438953483}Prev(){vfld12232066859008}`, `{vfld137438953483}Next(){vfld12232066859008}`, and `{vfld137438953483}Exit(){vfld1044979707918942208}`.
29. Save your document file, and compile and test as usual. You get an error message *Table formatting too complex* that can be ignored. (It is explained under the Comment section below.) The window should look like Figure 6-13.

```
{ewc vwrht2, TsTextButton, "Figure  
6½13"[Macro=JI('viewer.mvb>SecWin', `fig6_13')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

30. Use the new controls to switch between the two topics, and to exit Viewer.

### How It Works

Just as in How-To 6.5, you created the button bar using the techniques demonstrated in How-To 4.2—a table with ewc commands. Each command is set up to execute an appropriate Viewer command if the user clicks on the picture.

The button bar is displayed in a regular pane through the PaneID command included in the Config script. The commands in this script are executed when the application is loaded.

### Comment

The `{vfld2305858952132296716}Table formatting too complex{vfld2305857852620668928}` message results from setting the `{vfld137438953484}cell height{vfld4211991010332377088}`. Viewer does not support this command. Fortunately, as you saw, Viewer sets



the correct height for the table cells automatically.

The proper size and position of the pane containing the controls is determined by trial and error, or from knowing the size of the pictures.

Regular panes are automatically resized to fit the topic being displayed. As a result, picture bars such as in this How-To must be carefully designed, and the panes properly positioned, for the best appearance. This resizing is also part of the reason why the right and bottom portions of the pane have a different color. The background color of the window is showing through in these areas because it is not covered by a pane. The window

{vfld137438953484}background color{vfld11132555231232} does not default to using the default color—that option must be set. As you see in this demonstration, it is important to make sure the

{vfld137438953482}background colors{vfld280933810831360} of the window and all panes are consistent, or at least compatible, when using panes. This is not an issue if you don't use panes, because the master pane covers the entire window.

The {vfld137438953484}border{vfld-9007337234860343296} around the controls pane is eliminated because a border around a resized pane (when the topic is smaller than the space provided) is distracting rather than helpful.

## Create a Graphic Interface in a Floating Window?

Complexity: INTERMEDIATE

### Problem

How-To 6.6 looked really good, but now I want to try the same thing with the controls in a floating window.

### Technique

You use the same techniques of creating a table with ewX commands as in the two previous sections, but this time display the topic containing the controls in a secondary window.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP6\_7, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP6\_7. You create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the picture files for this How-To from directory VIEWERHT\HOWTOS\CHAP6\CHAP6\_7\PICTURES on the CD-ROM to the VIEWERHT\CHAP6\_7\PICTURES subdirectory on your hard disk.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP6\_7 directory. Enter the name of your document file as TEXT\CHAP6\_7.RTF.

Next define the secondary window and a topic group:

4. Choose Title Options from the Section menu, and enter contents in the Contents Topic field.
5. Choose {vfld137438953482} Window Definitions {vfld1477325272146509824} from the Section menu. Click on the New button to create a new window, then the Properties button to display the Window Properties dialog box.
6. Change the Window Name to SecWin, and the Window Caption to Secondary Window. Change the Stay on Top field to Yes. Change the Minimize with Main field to Yes, change Width to 1024, and change Left to 0. Use the System Control box to close the Window Properties dialog box, and click on OK in the Window Definitions dialog box.
7. Choose Groups from the Section menu to display the Groups dialog box, and click on the New button.
8. Clear the Searchable check box. Leave the name as the default Group1. Click on OK.

Next create the topics for testing:

9. Use Project Editor to start Word and create your document, and create a new topic with context string contents. Be sure to delete the page break that is created before your topic.
10. Use Topic Editor to define a Topic Group footnote for Group1 and Sequence Number 010.
11. Use Topic Editor to define a Topic Entry Command (! footnote). Click on the Paste Command button, and select the JumpID command.
12. Click on the Edit Command button, and change the TitleFile field to `chap6\_7.mvb`. Change the WindowName field to `SecWin` by using the pull-down list, and change the Context field to `controls`. Be sure to include the proper left and right single quotation marks in each field. Click on OK twice to return to the document.
13. Enter the following text:

This is the Contents topic. Use the Next button to display the next topic.

14. Insert a blank line and create a new topic with context string topic2. Define a Topic Group footnote for Group1 and sequence 020. Enter the following text:

This is the second topic. Use the Previous or Contents buttons to display the other topic.

Next create the topic containing the controls:

15. Insert a blank line and create a new topic with context string controls.
16. Choose Insert Table from the Table menu, and create a table with four columns, one row, and a 1.6-inch column width.
17. Select the entire row of the table, then choose Column Width from the Table menu, and set Space Between Columns to 0. Click on OK.
18. Position the insertion point in the first table cell, and use Topic Editor to insert a Picture (Using ewX) command. Select the PICTURES\CONTENTS.BMP file, and check the Store Picture in Baggage check box.
19. Click on the Paste Command button, and select the JumpID() command. Click on the Edit Command button, and change the TitleFile to `chap6\_7.mvb`, WindowName to `main`, and Context to `contents`. Click on OK twice to return to the document.
20. Repeat steps 18 and 19, inserting picture files PREVIOUS, NEXT, and EXIT with corresponding commands Prev(), Next(), and Exit(). These commands do not need to be edited.
21. Save your document file, and compile and test as usual. The window should look like Figure 6-14.

```
{ewc vwrht2, TsTextButton, "Figure  
6½14"[Macro=JI(`viewerht.mvb>SecWin', `fig6_14')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

22. Try to use the new controls to switch between the two topics. Switch to the second topic by using the standard Browse >> button, and return by using the new Contents picture control. Try minimizing and restoring the main window. You should find that the new Previous and Next controls don't work. Exit by clicking on the new exit picture.

### **How It Works**

The JumpID Topic Entry command in the contents topic, which you created in steps 11 and 12, creates a secondary window that was previously defined in Project Editor, and displays the topic with context string controls. You could just as easily have inserted this command into the project file's Config script as you did in How-To 6.6. However, it is generally a good idea to avoid adding commands to the Config script when secondary windows are used, as these commands are executed both when the file is loaded, and each time a secondary window is displayed. Re-executing some commands is harmless, but others could cause unexpected topics to be displayed or other problems.

A different command is associated with the Contents picture—JumpID instead of `{vfld137438953483}Contents(){vfld13331578486784}`—because Viewer ignores a `{vfld137438953484}Contents(){vfld13331578486784}` command executed in a secondary window. The same thing is true of the `{vfld137438953484}Prev(){vfld13331578486784}` and `{vfld137438953484}Next(){vfld-9151314983982727168}` commands, but there aren't any substitute commands that would work with any topic and group structure. This is why the Previous and Next controls don't work. The `Exit()` command does work in a secondary window.

### **Comment**

A floating window interface can be created with some programming in either C or Visual Basic. The programming requires issuing appropriate standard commands to Viewer through a program interface when the user clicks on a picture, and using the Viewer extensions capability to follow when the Viewer window is minimized, maximized, or exited. The VBX custom control included on the enclosed CD-ROM is required to write this program in Visual Basic. The design of program control of a Viewer application is demonstrated in Chapter 10.

The wide spacing between the buttons is caused by the table that was used. How-To 4.2 demonstrates the effect of various techniques for creating button bars.

You could use the JI command in place of the JumpID command. Abbreviated command names work exactly the same as full names.

## Create a Graphic Navigation Bar?

Complexity: DIFFICULT

### Problem

I want to create a graphic control bar that lets the user view a selected section of the application. The graphics must reflect which section the user is in at all times. I don't want to keep the standard button bar.

### Technique

You create a regular pane above the master pane, and display the controls in a table in that pane. Different topics, using different pictures, are displayed based on the current section.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP6\_8, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP6\_8. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the picture files for this How-To from directory VIEWERHT\HOWTOS\CHAP6\CHAP6\_8\PICTURES on the CD-ROM to the VIEWERHT\CHAP6\_8\PICTURES subdirectory on your hard disk.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP6\_8 directory. Enter the name of your document file as TEXT\CHAP6\_8.RTF.
4. Choose Title Options from the Section menu, and enter contents in the Contents Topic field.

Next define the pane:

5. Choose Window Definitions from the Section menu, and click on the Properties button to display the Window Properties dialog box. Click on the Master Pane button to display the Master Pane Properties dialog box.
6. Clear the Auto-position check box, then click on the Preview On button.
7. Drag the borders of the master pane to expose an area above the pane. Double-click on the System Control box to close this window. Close the Window Properties dialog box the same way.
8. In the Window Definitions dialog box, click on the Panes file folder tab, then click on the New button. Click on the Properties button to display the Pane Properties dialog box.
9. Change the Pane Name field to Controls.
10. Change the Dismiss When field to Title is Closed, and the Border field to (none).
11. Click on the Windows button to display the Pane Associations dialog

box, then check the Show in Window check box.

12. Click on the Preview button to display the window layout, and then drag and stretch the Controls pane to cover the area above the master pane, as shown in Figure 6–15.

```
{ewc vwrht2, TsTextButton, "Figure  
6½15"[Macro=JI('viewerht.mvb>SecWin', 'fig6_15')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

13. Use the System Control box to close the preview window and the Window Properties dialog box, then click on OK in the Window Definitions dialog box.

Next create three topic groups with entry scripts:

14. Choose Groups in the Section menu to display the Groups dialog box, and click on the New button.
15. Clear the Searchable check box. Leave the name as the default Group1.
16. Click on the Entry Script button, then click on the Paste Command button, and select the `{vfld137438953483}PaneID{vfld1477325272146509824}` command.
17. Click on the Edit Command button. Enter `qchPath` in the TitleFile field without quotation marks, and enter a single space in the WindowName field without quotation marks. Enter ``control_1'` in the Context field, and ``Controls'` in the PaneName field with single quotation marks for both fields. Enter 0 in the PrintTabCopyOrder field, and click on OK twice.
18. Repeat steps 14 through 17 to create Group2 and Group3. Use context strings ``control_2'` and ``control_3'`. Note that a bug in Project Editor sometimes causes problems when defining Entry or Exit Scripts. This appears as two occurrences of each group, with no scripts. It is visible after exiting the Groups dialog box and reselecting that menu item. If this happens, delete the first occurrence of each group and redefine the Entry Scripts.
19. Choose Config from the Section menu entry. Remove the Std20Buttons() script entry by selecting that line and pressing **[DEL]**. Click on OK. Save the project file by choosing Save from Project Editor's File menu.

Next prepare the document. First create the topics for testing:

20. Use Project Editor to start Word and create your document, and create a new topic with context string contents. Be sure to delete the page break that is created before your topic.
21. Use Topic Editor to define a Topic Entry command. Click on the Paste Command button, and select the PaneID command. Click on OK.
22. Click on the Edit Command button. Enter `qchPath` in the TitleFile field without quotation marks, and enter a single space in the WindowName field without quotation marks. Enter ``control_0'` in the Context field, and ``Controls'` in the PaneName field with single quotation marks for

both fields. Enter 0 in the PrintTabCopyOrder field, and click on OK twice to return to the document.

23. Enter text:

This is the Contents topic. Please click on the desired section.

24. Create a new topic with context string Sect\_1A. Define a Topic Group footnote for Group1 and Sequence Number 010.

25. Enter text:

This is the first topic in section 1.

26. Repeat steps 24 and 25 twice, creating topics with context strings Sect\_1B and Sect\_1C, Sequence Numbers 020 and 030, and appropriate text.

27. Repeat steps 24 to 26 twice, creating sections 2 and 3. Use appropriate group and context string names and text.

Next create the topics containing the picture controls:

28. Create a new topic with context string control\_0.

29. Choose Insert Table from Word's Table menu, and create a table with one row and three columns, and a column width of 1.5 inches.

30. Choose Column Width from the Table menu, and set the Space Between Columns to 0.

31. Position the insertion point in the first table cell, and use Topic Editor to define a Picture (Using ewX) command. Click on the Options button.

32. Select the file PICTURES\FIRST.BMP, and check the Store Picture in Baggage check box. Click on the Paste Command button, and select the JumpID command. Click on the Edit Command button, and change the TitleFile field to qchPath, WindowName to one space, and Context to `sect\_1a'. Click on OK until you are back in the document.

33. Repeat steps 31 and 32 with the two remaining cells. Use files SECOND.BMP and THIRD.BMP, and jump to context strings sect\_2a and sect\_3a.

34. Repeat steps 28 through 33. The topic should have context string control\_1. The first table cell should display file FIRSTB.BMP, and not have any associated command. The other two cells should be identical to those in the control\_0 topic.

35. Repeat steps 28 through 33. The topic should have context string control\_2. The second table cell should display file SECONDB.BMP, and not have any associated command. The other two cells should be identical to those in the control\_0 topic.

36. Repeat steps 28 through 33. The topic should have context string control\_3. The third table cell should display file THIRDB.BMP, and not have any associated command. The other two cells should be identical to those in the control\_0 topic.

37. Save the document, and compile and test the application as usual. The results should look like Figure 6–16.

```
{ewc vwrht2, TsTextButton, "Figure  
6i½16"[Macro=JI('viewerht.mvb>SecWin', `fig6_16')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

Each topic group is defined with an entry script containing a `PaneID` command. This command displays the picture controls appropriate to that group of topics in the regular pane. You could also have defined exit scripts, but none were needed in this How-To.

You remove the `{vfld137438953483}Std20Buttons(){vfld-9007337234860343296}` command to eliminate the standard button bar. Unfortunately, this eliminates the browse buttons and you didn't include your own in the graphic controls. As a result, you can get to the first topic in each group, but not to the others! You also can't get back to the Contents topic, or retrace your steps through the History function. This shows the importance of planning before you replace any of the standard menus or buttons.

You defined a topic-entry script in the Contents topic that displays the initial picture controls. This can be done through the Config section of the project file with the same effect.

The sets of picture controls are nearly identical. Tables are used to assure that each picture is in the same position in each topic, to avoid the appearance of pictures jumping around as the user changes topics. A grayed-out picture is substituted in each case for the current topic group, to indicate that this is the current selection. No command is associated with those pictures because you are already in that topic group. The usual command could have been left in place, to provide for returning to the beginning of the group.

### Comment

Did you notice how long it took to display the picture controls when the file was loaded? The use of regular panes displaying different topics results in significant added overhead in Viewer. This would be even slower if the file being viewed were on a CD-ROM disk—their slower operation would take more time to retrieve the additional topic. This is less apparent in the later topics because the previous appearance is consistent. The appearance of that pane initially is jarring because it shows the Program Manager or other previous underlying material.

When you define `PaneID`, `JumpID`, or similar commands and want to display a topic from the current file in the main window, you can use `{vfld137438953482}qchPath{vfld-9007199795906871296}` for the Title File and a single space for the Window Name. The `qchPath` entry is a special Viewer-defined variable that contains the path and file name of the active Viewer MVB file. This variable, with a blank Window name, describes the default case of "main window in the current file." The `qchPath` variable cannot be used with a nonblank window name because of the command syntax. In that case the actual file name, such as `CHAP6_8.MVB`, must be hard-coded in the command.

The proper size and position of the pane containing the controls is determined by trial and error, or from knowing the size of the pictures.



Regular panes are automatically resized to fit the topic being displayed. As a result, picture bars such as in this How-To must be carefully designed, and the panes must be properly positioned, for the best appearance. This resizing is also part of the reason why the right and bottom portions of the pane have different colors. The background color of the window is showing through in these areas because it is not covered by a pane. The window background color does not default to using the default color—that option must be set. As you see in this demonstration, it is important to make sure the background colors of the window and all panes are consistent, or at least compatible, when using panes. This is not an issue if you don't use panes, because the master pane covers the entire window.

The border around the controls pane is eliminated because a border around a resized pane (when the topic is smaller than the space provided) is distracting rather than helpful.

A value of 0 for `PrintTabCopyOrder` prevents the topic from being printed or copied.

## **Browse Within a {vfld137438953482}Secondary Window{vfld3800470531342336}?**

Complexity: INTERMEDIATE

### **Problem**

I'm going to use a secondary window to display a series of topics, just as though it is the main window. How can I give my users a Browse function that works within the secondary window?

### **Technique**

You can't create buttons within a secondary window, so use hot spot pictures again. Put these pictures in a non-scrolling region within each topic, and create pictures for the Contents and Exit functions.

Just for effect, reduce the main window to an icon right away, and do everything in the secondary window.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### **Steps**

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP6\_9, and the standard subdirectories—TEXT, SOUNDS, PICTURES, MOVIES—under CHAP6\_9. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the picture files for this How-To from directory VIEWERHT\HOWTOS\CHAP6\CHAP6\_9\PICTURES on the CD-ROM to the VIEWERHT\CHAP6\_9\PICTURES subdirectory on your hard disk.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP6\_9 directory. Enter the name of your document file as TEXT\CHAP6\_9.RTF.

Next define the secondary window:

4. Choose Window Definitions from the Section menu, and click on the New button to define a new window.
5. Click on the Properties button to display the Window Properties dialog box. Change Window Name to SecWin, and Window Caption to How-To 6.9. Be sure to leave the Minimize with Main field set to No. Set Height and Width to 700. Use the System Control box to close this dialog box. Click on OK in the Window Definitions dialog box to return to the main Project Editor window.
6. Choose Title Options from the Section menu, and enter contents as the Contents Topic. Click on OK.
7. Save the updated project file.

Next create the document file, and the only topic that is displayed in the main

window:

8. Use Project Editor to start Word and create your document, and create a new topic with context string contents. Be sure to delete the page break that is created before your topic.
9. Use Topic Editor to define a Topic Entry command script. Click on the Paste Command button, select the JumpID command, and click on OK. Click on the Edit Command button, and change TitleFile to `CHAP6\_9.MVB`, WindowName to `SecWin`, and Context to `sec\_contents`. Be sure to use the proper left and right single quotation marks around each entry. Click on OK.
10. Move the insertion point to the next line in the command script, then click on the Paste Command button again. Select the PositionTopic command, then click on OK.
11. Click on the Edit Command button, and change WindowName to `main`, X and Y to 0, Width and Height to 100, UnitFlag to 0, and WindowState to 2. Click on OK.
12. Move the insertion point to the next line in the command script, then click on the Paste Command button again. Select the FocusWindow command, then click on OK.
13. Click on the Edit Command button, and change the WindowName to `SecWin` by using the pull-down list. Click on OK twice to return to the document.
14. Enter the following text:  
This is the Contents topic of the main window.

Now create the topics that display in the secondary window.

15. Create a new topic with context string sec\_contents.
16. Choose Paragraph from Word's Format menu, and check the Keep With Next option. Click on OK.
17. Choose Insert Table from the Table menu, and create a table with one row and four columns, with a 1.5-inch column width.
18. Position the insertion point in the first table cell, and use Topic Editor to define a Picture (Using ewX) command. Select file PICTURES\CONTENTS.BMP, and check the Store Picture in Baggage check box.
19. Click on the Paste Command button, select the JumpID command, and click on OK.
20. Click on the Edit Command button, and change TitleFile to `chap6\_9.mvb`, WindowName to `SecWin`, and Context to `sec\_contents`. Click on OK twice to return to the document.
21. Repeat steps 18 through 20 for the remaining cells. Use files PREVB.BMP and NEXTB.BMP in the second and third cells with no commands. Use EXIT.BMP in the last cell with the Exit() command. This command has no parameters and so does not need to be edited.
22. Insert one blank line and turn off the Keep with Next formatting. Enter text:  
This is the Contents topic of the secondary window.

23. Insert a few blank lines, and enter the text `Jump to next topic`. Select that text, and invoke Topic Editor to create a text hot spot.
24. Select the Jump to element, and enter `topic_1` as the Context String, and `SecWin` as the Window. Leave the MVB Filename blank. Click on OK to return to the document.
25. Create a new topic with context string `topic_1`. Repeat steps 16 through 22, except use file `NEXT.BMP` in the third cell, with a `JumpID` command specifying `TitleFile `chap6_9.mvb'`, `WindowName `SecWin'`, and `Context `topic_2'`. Use the text `This is the first topic`.
26. Create a new topic with context string `topic_2`. Repeat step 25, except the second cell must use file `PREV.BMP` and a `JumpID` command specifying `TitleFile `chap6_9.mvb'`, `WindowName `SecWin'`, and `Context `topic_1'`. Also, the `JumpID` command in the third cell must reference context string `topic_3`. Use the text `This is the second topic`.
27. Create a new topic with context string `topic_3`, like the previous topic. The command in the second cell must reference context string `topic_2`. The third cell must use file `NEXTB.BMP` and no command. Use the text `This is the third (and last) topic`.
28. Save your document file, and compile and test as usual. The results should look like Figure 6–17.

```
{ewc vwrht2, TsTextButton, "Figure
6i½17"[Macro=JI('viewerht.mvb>SecWin', `fig6_17')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

The secondary window is defined through Project Editor in the usual way. The most important point in this definition is leaving the `Minimize With Main` option set to `No`. This is changed to `Yes` for secondary windows that operate with the main window.

The MVB file's Contents topic performs only one function—executing a topic-entry command script that displays the secondary window and minimizes the main window. These commands cannot be placed in the `Config` section of the project file because that script is re-executed when a secondary window is displayed. This would cause an infinite loop of executing the same commands. The first command is a `JumpID` that specifies the secondary window. This creates the window and displays the initial topic in it. The second command is a `PositionTopic` that is executed to minimize the main window. This command requires entering the size and position of the window. The third command is a `FocusWindow`, which makes the secondary window the active window.

The pictures are placed within the topics, rather than defining them in separate topics displayed in a regular pane, because the pictures and

commands must be changed for every topic. By including them within the topics the proper pictures and controls are always displayed automatically. This eliminates the need to execute a command to display the pictures separately, and avoids the time that would take.

The appropriate pictures are replaced with grayed versions to reflect the beginning or end of a browse sequence, just as the standard buttons are grayed out in the main window. There is no way to detect this automatically—the desired pictures must be defined in each topic. The pictures used for the Contents and Exit functions remain the same in all topics, as they are never inactive.

The commands associated with the browse pictures also change with each topic. The Previous picture does not have a command when the first topic in the sequence is displayed, and the same is true of the Next picture in the last topic in the sequence. In all other cases, the pictures execute JumpID commands that specify the context string of the previous or next topic in the sequence. All of this is hard-coded—it cannot be determined automatically. The Prev() and Next() commands do not work in secondary windows. The actual file name must be specified in the TitleFile parameter because a window name is being used. The command syntax does not permit the general qchPath variable to be used with a window name.

The pictures are displayed in a table with the Keep with Next paragraph format option. The table assures that the pictures remain in the same positions between topics, preventing the appearance of the pictures jumping around as the user changes topics. The formatting defines this portion of each topic as a non-scrolling region, so the pictures remain visible even if the topic contents are scrolled by the user.

### **Comment**

Note that this application fails if the name of the MVB file is changed, because the name must be hard-coded into the commands. If necessary, this restriction can be avoided by writing a program to provide an external command. That command could determine the file name while executing.

A Windows API function could have been called to minimize the main window without requiring the window size and position. In this case there was no value to the extra effort to define the desired external function. Viewer's `{vfld137438953483}CloseWindow{vfld-9007337234860343296}` command could not be used to eliminate the main window because that would also close the secondary window.

The wide spacing between the buttons is caused by the table that was used. How-To 4.2 demonstrates the effect of various techniques for creating button bars.

The hard-coding of context strings in the commands can cause errors as the application is revised. The commands in two topics must be updated whenever a topic is added or removed. Careful documentation of the topic context strings and references is especially critical when this technique is used.

## 6.10 Tips and Tricks

- ⇒ Commands can be copied from one table cell to another. They can then be edited in the document, or by selecting the entire command and invoking Topic Editor. If you edit the file name in the document, be sure to add the new file to Baggage if necessary. This can be done in the Project Editor main window by selecting the Baggage file folder tab and using the Edit menu. If you edit the file name through Topic Editor, it automatically unchecks the Store Picture in Baggage check box—be sure to turn it back on. Topic Editor also asks if the previous file should be removed from Baggage. Under these circumstances you should click on the No button.
- ⇒ Buttons, menus, and accelerator keys are the easiest interface elements to modify. Viewer provides commands to add, delete, and redefine these elements, and these commands can be easily executed within topic-entry or group-entry scripts. If you need to change the commands executed by part of your interface frequently, you should try to use these redefinable elements. If you can't, you should see if an external command could simplify the procedure. This can make design options for your interface practical that otherwise might be difficult to implement.
- ⇒ Although Viewer does provide many opportunities for designing a custom interface without special programming, programming may provide improved speed or design simplicity. The Viewer documentation provides sufficient information for an experienced programmer to write any external commands required. Most programs that may be required can be written in Visual Basic with the VBX controls included in the enclosed CD-ROM disk. This makes such programming easier for many people. Experienced programmers can also be located and hired through the Viewer section in CompuServe.
- ⇒ The {vfld137438953482}window border{vfld11132555231232}, {vfld137438953482}caption{vfld11132555231232}, and {vfld137438953482}Min{vfld11132555231232} and {vfld137438953482}Max{vfld-541165879296} buttons can be eliminated by calling the appropriate Windows API functions. For example, the following commands can be added to your Config script to eliminate the Max button and resizable border:

```
RegisterRoutine("user","SetWindowLong","UiU")
RegisterRoutine("user","SetWindowPos","Uuiiiu")
{vfld137438953483}SetWindowLong{vfld11132555231232}
({vfld137438953482}hwndApp{vfld3131967461654528}, -16,
0x16CA0000)
{vfld137438953483}SetWindowPos{vfld-35184913254711296}
(hwndApp, 0, 0, 0, 0, 0, 39)
```

To also eliminate the caption and Min button, use the following SetWindowLong call instead:

```
SetWindowLong(hwndApp, -16, 0x16080000)
```

These commands are documented in the Windows Software Development Kit (SDK) and some commercial books. These functions

should only be used by, or on the advice of, experienced Windows programmers. Errors in the parameter values can cause major problems.

- ⇒ When defining file names with paths inside commands, all  
{vfld137438953482}backslashes {vfld13331578486784} must be  
doubled because single  
{vfld137438953484}backslashes {vfld280933810831360} are considered  
to be RTF commands. For example, use EP("c:\\work\\myprog.exe"). If  
the command is nested inside another command, use quadruple  
backslashes. For example: CBB("my\_button", "EP(`c:\\\\work\\\\  
myprog.exe`)")

**Edit Command**

**Edit Command:** CreateButton

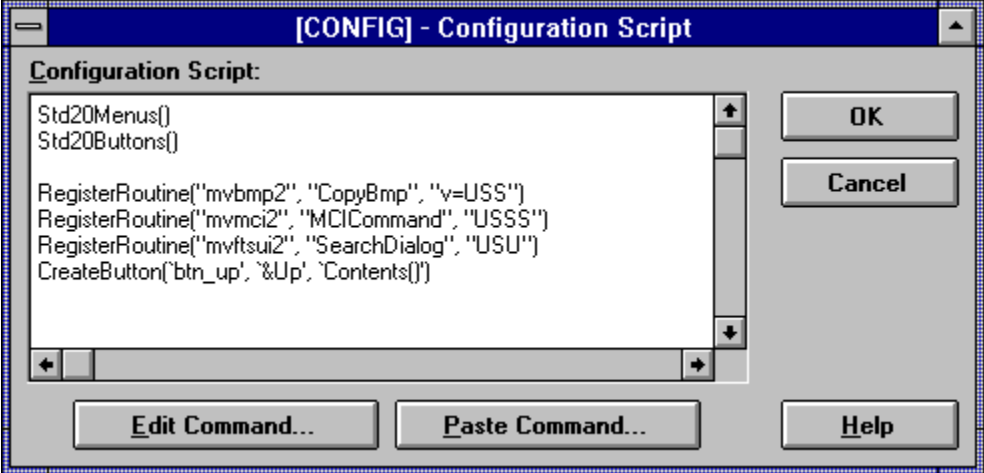
**ButtonID:**

**ButtonCaption:**

**Command:**

ButtonID: Specifies the identifier for the button





Microsoft Word - CHAP6\_1.RTF

File Edit View Insert Format Tools Table Window Help

Normal Times New Roman 0

1 Chapter 3: Create a Simple Application

1

1

.....Start a New Projectsect\_3\_1

.....Create directoriessect\_3\_1\_a

.....Create Project Filessect\_3\_1\_b

.....Create a Contents Topicsect\_3\_2

.....Create the Document Filesect\_3\_2\_a

.....Create the Footnotessect\_3\_2\_b

---

Footnotes Close

#chap 3

!CBB("bin\_up","Contents()");ExableFullun("bin\_up");EB("bin\_up")

#sect 3 1

!CDD("bin\_up","JumpID(qchPact,"chap 3");ED("bin\_up")

#sect\_3\_1\_a

!CDD("bin\_up","JumpID(qchPact,"sect\_3\_1");ED("bin\_up")

#sect\_3\_1\_b

Pg 9 Sec 1 9/15 At \_n Col 1 100% NUM

**Edit Command**

**Edit Command:** AppendItem

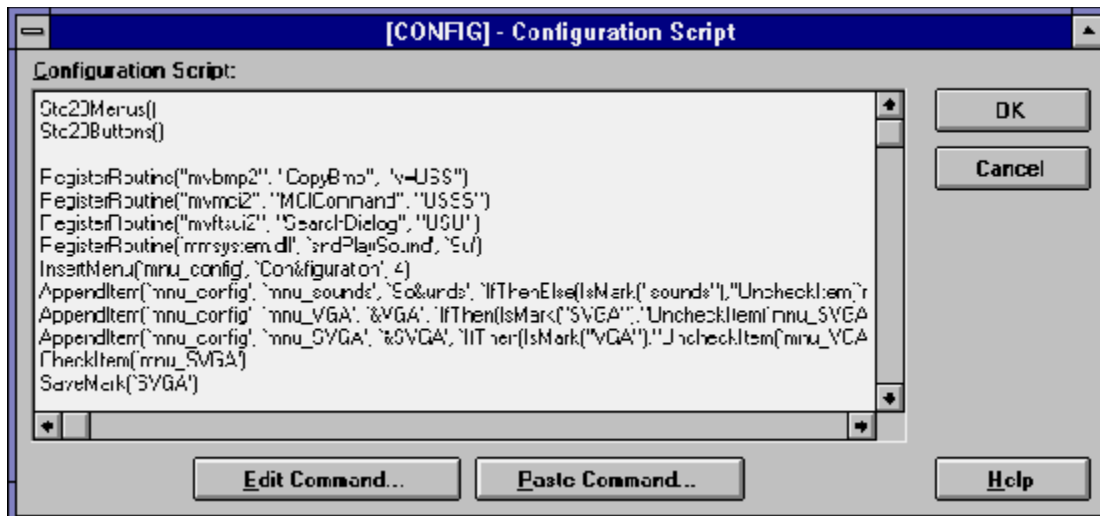
**MenuID:**

**NewItemID:**

**ItemCaption:**

**Command:**

MenuID: Specifies the identifier for the menu



Microsoft Word - CHAP6\_2.RTF

File Edit View Insert Format Tools Table Window Help

Normal Times New Roman 0

---

#!¶  
 ¶  
 Play-sound{ctx\_sounds}¶  
 Show-picture{ctx\_picture}¶  
 ¶

---

#!¶ This plays a times if allowed. Click on the GO BACK button to return. ¶

---

#!¶  
 ¶  
 {ewc MVBMF2,ViewerBmp2,[caption="This is a Super VGA Picture"]pictures\svga.bmp}¶  
 ¶

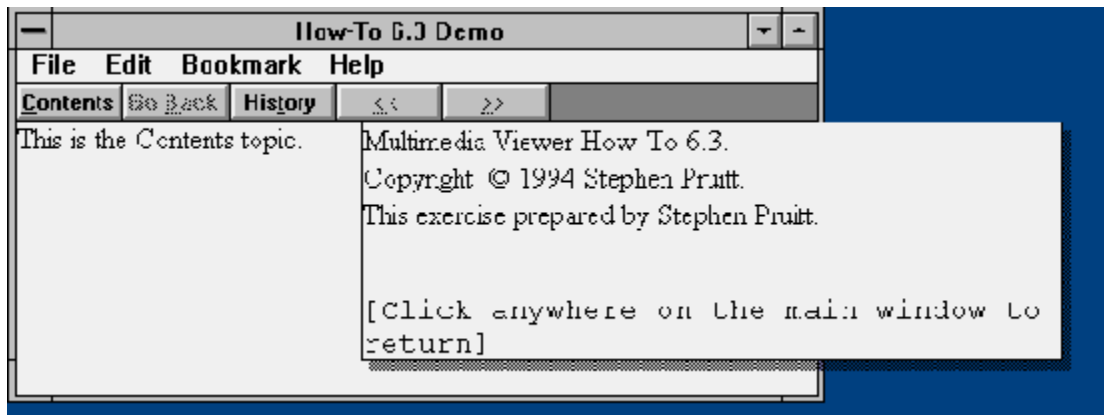
---

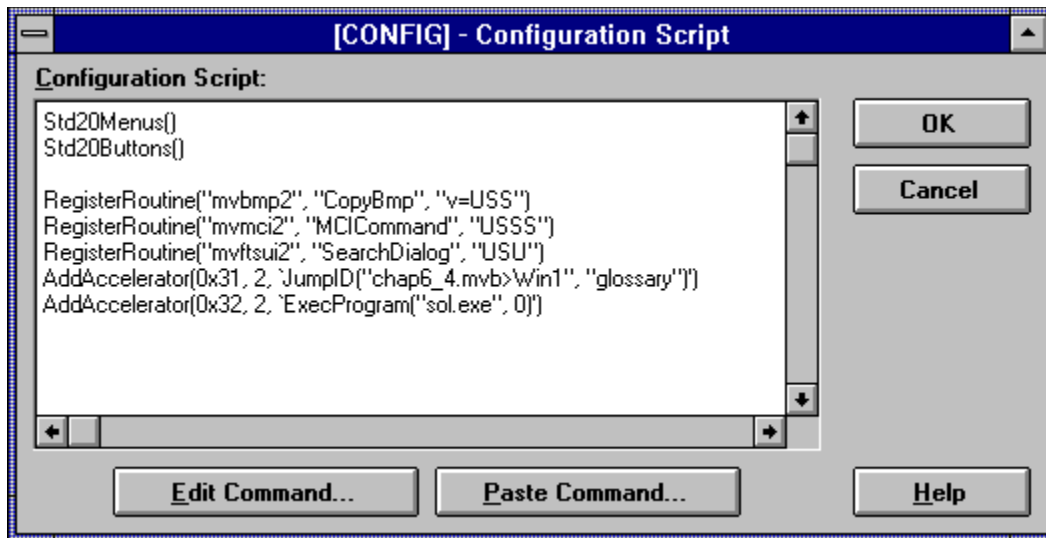
Footnotes Close

#content{¶  
 #ctx\_sounds{¶  
 !IfThen(IsMark("sounds"),"sndPlaySound("sounds\times.wav",1)")¶  
 #ctx\_picture{¶  
 !IfThen(IsMark("VGA"),"jump||XgchPath,ctx\_vga")¶  
 #ctx\_vga{¶

---

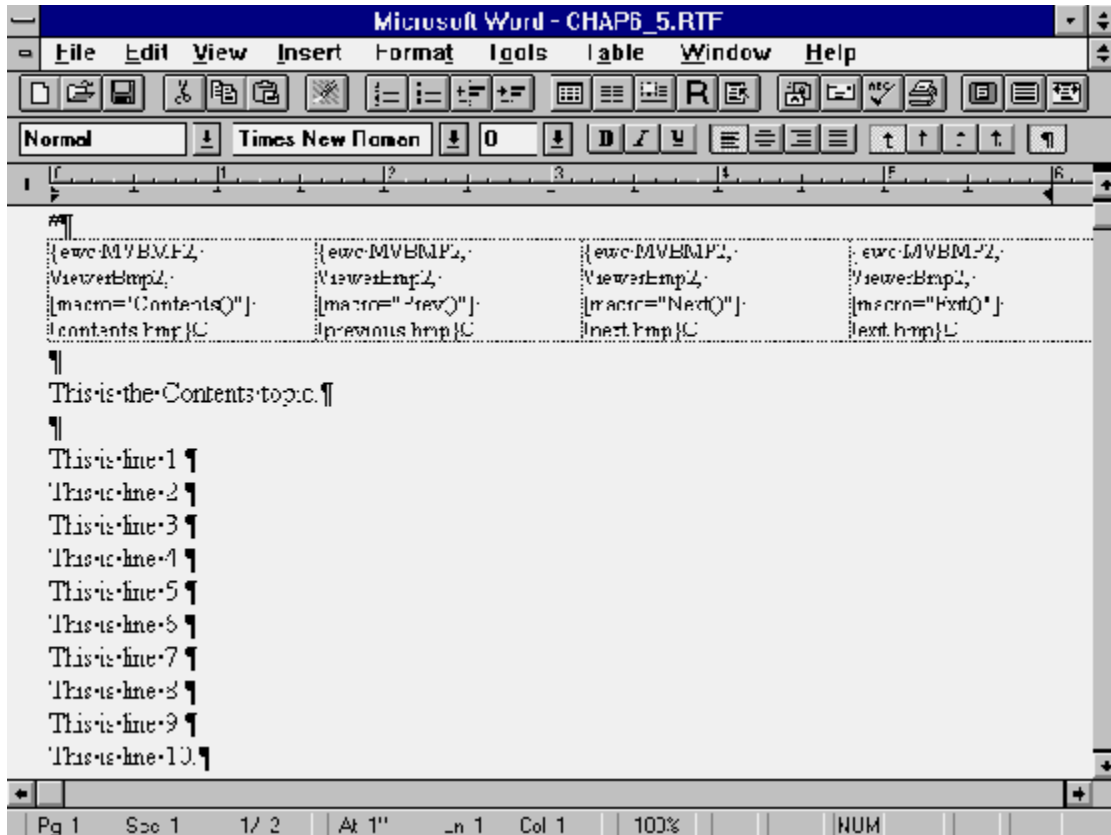
Pg 1 Sec 1 1 / 4 At \_n Col 1 100% NUM














How-To 6.5 Demo


File Edit Bookmark Help

Contents Go Back History << >>

 Contents

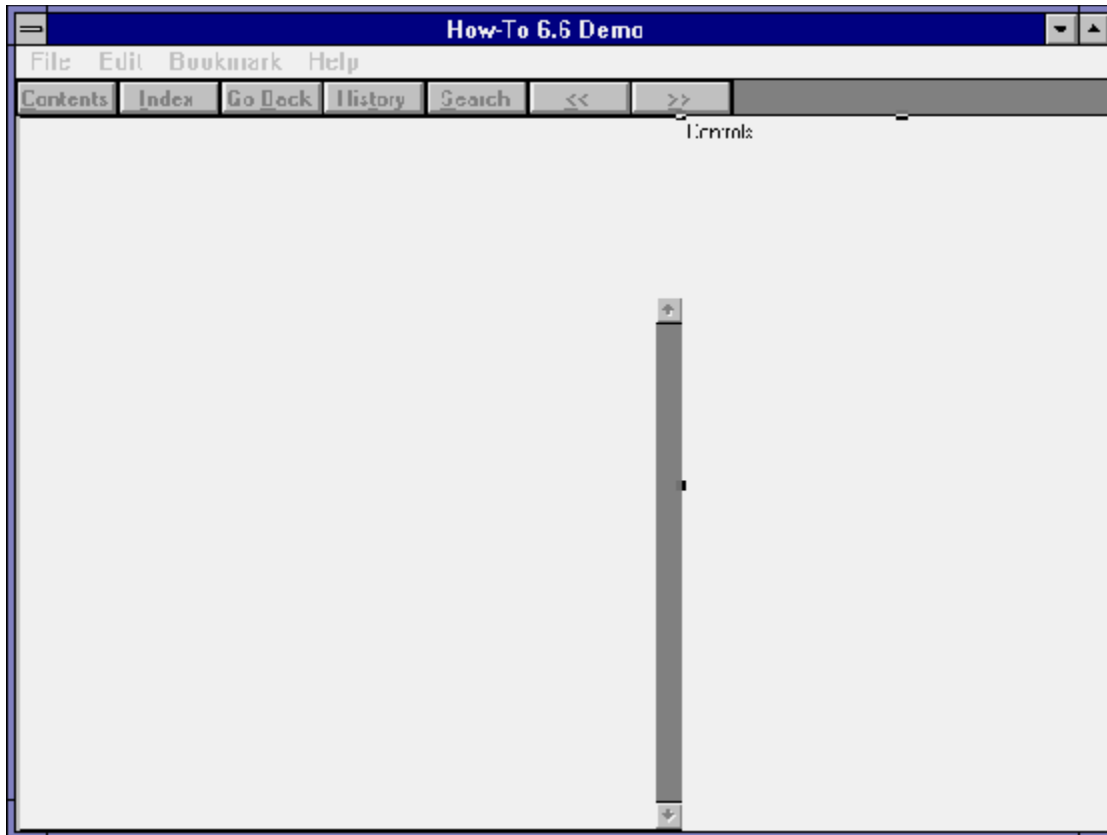
 Previous

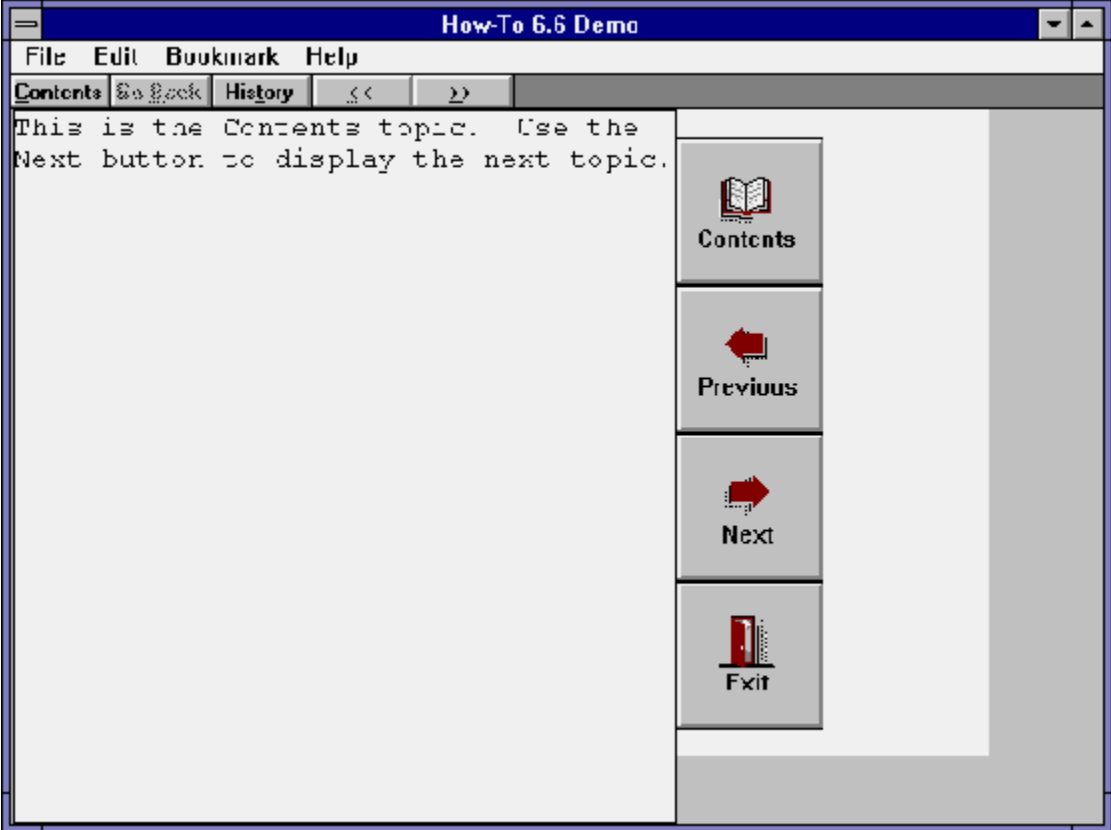
 Next

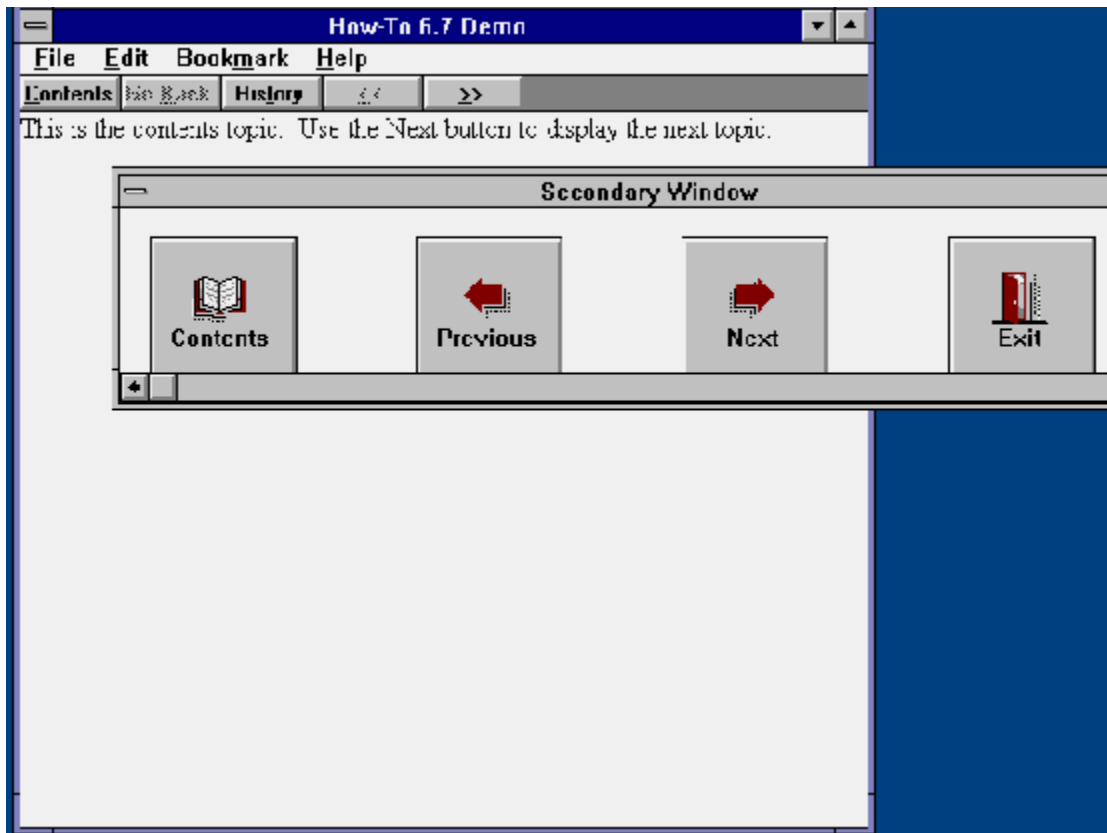
 Exit

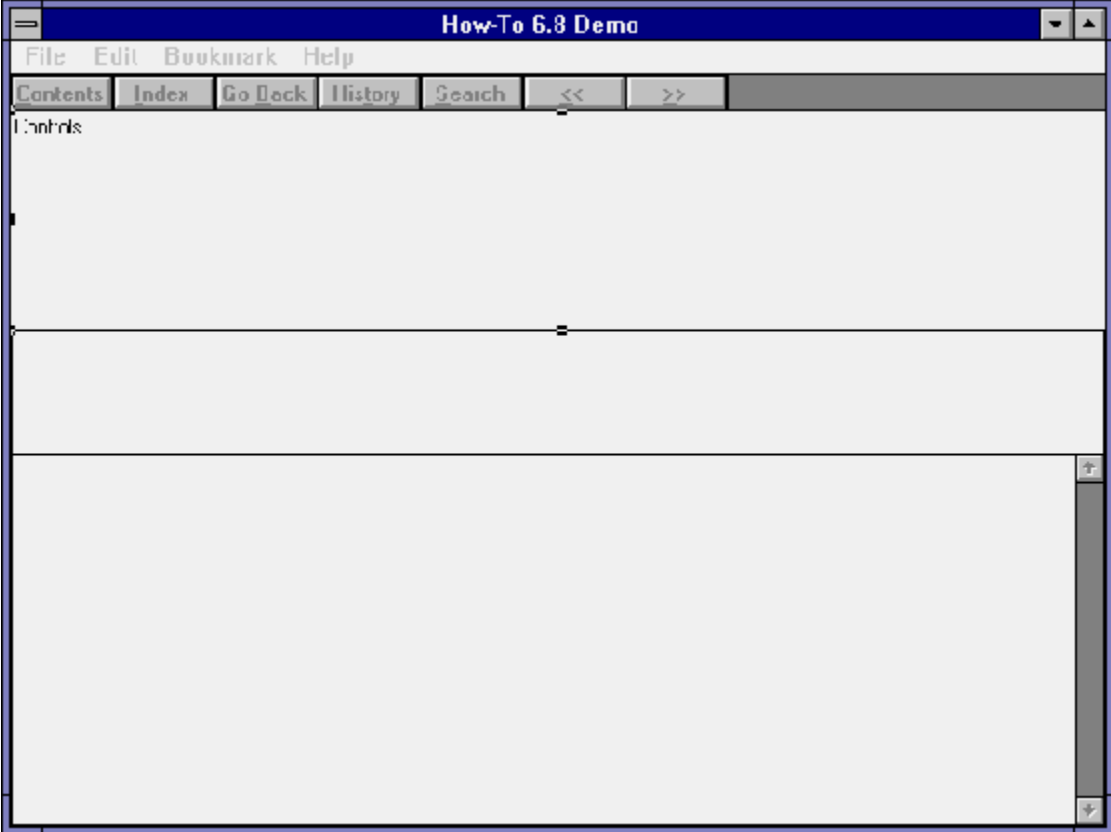
This is line 26.  
This is line 27.  
This is line 28.  
This is line 29.  
This is line 30.

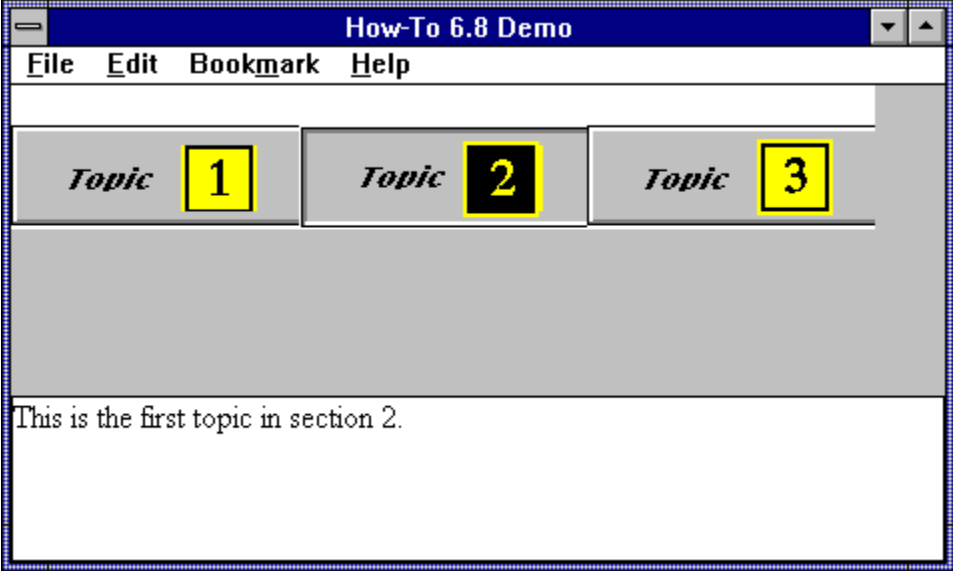
[Go to another topic](#)













This is the third (and last) topic.





Most Viewer applications are compiled into a single MVB file. The increasing popularity of CD-ROM drives, with their enormous capacity, makes it practical to distribute even very large applications in a single file. Despite that, some applications lend themselves to being divided into two or more files. Examples of when this might be more practical include the following:

- ü When part of the application information changes frequently, while the rest is stable. Dividing these portions into separate files might reduce the compilation time greatly—you don't need to recompile the stable portions.
- ü When two or more authors are independently developing portions of the application. Separating these portions could simplify the process of compiling and testing.
- ü When not enough disk space is available to compile the entire application into a single file. Chapter 2 explains the tremendous amount of space that a large application can require.

At this point you might ask “What's the big deal? I know how to code interfile jumps. I saw the field for the file name while I was doing the other How-Tos.” It's true that many parts of designing a multifile application are easy—the standard Back and History buttons let the user return to a different file, and Topic Editor lets you fill in a field with the name of the file for hot spots and jump commands. You don't even need to remember the syntax! Unfortunately, side effects can cause serious problems unless you prevent them.

One problem arises because the Contents button displays the contents topic of the current file by default. The command associated with this button must be modified so that the user can click on this button from the “other” file and have the proper topic displayed. How-To 7.1 demonstrates a simple solution to this problem.

A second problem may arise if you have a topic entry command in one file that jumps to a topic in the other file. What happens if the user then clicks on the Back button? Viewer redisplay the previous topic, which causes the topic entry command to be executed again. Oops! The user is right back where he or she started! One way you would run into this problem is if you wanted to define all your keywords in your primary file, so that your index is complete, even though some of these keywords should cause a topic in a secondary file to be displayed. The only way to do this is by defining keywords in a special topic in the first file. That topic would have a topic entry command that jumps to the desired topic in the secondary file. How-To 7.2 demonstrates a solution to this problem.

Another problem is that the standard search and index functions only display topics within the current file. This is a problem with most multifile applications. A multifile search can be developed based on the sample Search program, written in C, that is included with Viewer. This is not simple—it requires an experienced programmer. A multifile keyword index can be simulated with a multifile search by defining the desired terms as an author-defined data field in a search operation. This simulation can list the defined values if they are defined in a word wheel. Chapter 11 describes the standard search functions in detail.

All of these problems can be eliminated, or at least greatly reduced, by

your application design. For example, a secondary file that is only used to hold popup topics never presents these problems because the user can't jump to that file in the first place. A design with highly segregated parts, and no opportunity to jump between major areas, minimizes the opportunity for problems.

If you want to use multiple files because you have several authors working together, consider using multiple document files—one for each author. These files are eventually compiled together into a single MVB file. This is very easy to do. The Project Manager's Edit menu has items for inserting, changing, or deleting lines from the list of document files. Define the Contents topic carefully to be sure the right topic is used for that purpose. You should have one document file for the common front-end topics, and then divide up the rest of the project as desired.

This chapter doesn't show you how to solve all of these problems, but it does demonstrate some helpful basic techniques. How-To 7.1 demonstrates the basic operations involved, and How-To 7.2 shows a useful solution to the problems.

## Use Simple Interfile Operations?

Complexity: EASY

### Problem

I want to display topics in another MVB file through popups and jumps.

### Technique

You create the standard directories with two project files, two document files, and two MVB files. The file names have a suffix of *P* for the primary file and *S* for the secondary file.

(To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP7\_1 and the standard subdirectories—TEXT, SOUNDS, PICTURES, and MOVIES—under CHAP7\_1. Create these four directories for all projects, even if some are not needed.

Next create the primary files:

2. Use Viewer's Project Editor to create a new project file named CHAP7\_1P.MVP in your VIEWERHT\CHAP7\_1 directory. Enter the name of your document file as TEXT\CHAP7\_1P.RTF.
3. Choose Title Options from the Section menu to display the Title Options dialog box. Enter contents in the Contents Topic field. Click on OK.
4. Choose Config from the Section menu. Paste in a CBB command, then edit it. Select 'btn\_contents' from the pull-down list in the ButtonID field, and select Contents() in the Command field. Click on OK in each dialog box to return to Project Editor.
5. Save your project file by choosing Save from the File menu.
6. Use Project Editor to start Word and create your document, and create a new topic with context string contents. Be sure to delete the page break that is created before your topic.
7. Use Topic Editor to define a Topic Title of Primary contents.
8. Enter text as a heading line:

This is the Contents topic in the primary file.

9. Insert a few blank lines and enter the text:

Pop up from Secondary file

10. Select the text you just entered, and bring up Topic Editor to define a Hot Spot (text). Click on Popup as the Hot Spot Type, then select the Jump to... element.
11. Enter 'popup\_1s' in the Context field, leave the Window field blank, and enter 'CHAP7\_1S.MVB' as the MVB Filename.
12. Click on OK to return to the document.

13. Insert a couple of blank lines and enter the text:  
Jump to the secondary file

14. Select the text you just entered, and bring up Topic Editor to define a Hot Spot (text). Select the Jump to... element.

15. Enter `topic\_2s' in the Context field, leave the Window field blank, and enter `CHAP7\_1S.MVB' as the MVB Filename. Click on OK to return to the document.

16. Insert another couple of blank lines and enter the text:  
Jump to the secondary file via a topic entry command

17. Use Topic Editor to make this a hot spot jumping to context string topic\_2p in the current file.

18. Use Topic Editor to define a new topic with context string topic\_2p, and define a topic title of Primary topic 2.

19. Use Topic Editor to define a Topic Entry command, click on the Paste Command button, and select the JumpID command. Click on OK.

20. Click on the Edit Command button, then enter `CHAP7\_1S.MVB' in the TitleFile field. Enter a space in the WindowName field, and enter `topic\_2s' in the Context field. Click on OK twice to return to the document.

21. Enter the text:  
This is topic 2 in the primary file.

22. Save the document file and compile as usual.

Next create the secondary files:

23. Use Viewer's Project Editor to create a new project file named CHAP7\_1S.MVP in your VIEWERHT\CHAP7\_1 directory. Enter the name of your document file as TEXT\CHAP7\_1S.RTF.

24. Choose Config from the Section menu. Paste in a CBB command, then edit it. Select `btn\_contents' from the pull-down list in the ButtonID field, and select JumpID() in the Command field. Edit the JumpID parameters to read JumpID(`chap7\_1p.mvb', `contents'). Click on OK in each dialog box to return to Project Editor.

25. Save your project file.

26. Use Project Editor to start Word and create your document, and create a new topic with context string popup\_1s. Be sure to delete the page break that is created before your topic.

27. Enter the text:  
This message comes to you from the secondary file

28. Create a new topic with context string topic\_2s and topic title Secondary Topic 2. Enter text:  
This is topic 2 in the secondary file.

29. Save the document file and compile as usual.

Now test the files:

30. Choose Open from Project Editor's File menu to reopen the primary project file, then run Viewer to display the primary MVB file.
31. Click on the popup hot spot and see what happens.
32. Click on the first Jump to hot spot and see what happens. Click on the History button and see what is shown. Click on the Go Back button and see what happens. Click on the History button again and see what is shown.
33. Click on the second Jump to hot spot and repeat the procedure in step 32. Were you able to get back to the primary file? The History list should look like Figure 7-1.

```
{ewc vwrht2, TsTextButton, "Figure  
7-1"} [Macro=JI('viewerht.mvb>SecWin', `fig7_1')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

34. While in each file, click on the Contents button. See which topic is displayed.

### How It Works

In steps 9 through 11 you created a popup hot spot in the primary file that displays a topic from the secondary file. The only difference in procedure from defining a normal popup is entering a value in the MVB Filename field. In the document this appears as @CHAP7\_1S.MVB following the context string. This is the signal to Viewer that this is an interfile jump or popup. The jump hot spot created in steps 12 through 15 is created the same way, and looks very similar.

The jump in the third hot spot, created in steps 16 and 17, is a standard jump to another topic in the primary file. The second topic is created in steps 18 through 21. It contains a JumpID command that executes automatically when the topic is displayed. The JumpID causes a topic in the secondary file to be displayed. The result of these steps is that clicking on the hot spot displays topic *topic\_2p*, which in turn immediately displays topic *topic\_2s*.

When testing, the popup and the first jump hot spot should have performed exactly as you'd expect. Users would not be able to tell that two files were involved unless they looked at the History list. The list contains entries in the form *Filename: Topic Title*. Entries for the file currently being displayed show just the topic title, while entries for topics in any other files show the file name and topic title.

After clicking on the third hot spot and jumping to the secondary file, you should not have been able to return to the primary file by using the {vfld137438953484}Go Back {vfld-9151452422936199168} button. This button causes the previous topic, in the primary file, to be displayed—but that topic contains the topic entry command that jumps back to the secondary file. This problem is described in the introduction to this chapter. How-To 7.2 demonstrates a solution to it.

Clicking on the Contents button while in either file should have displayed the proper topic. You defined a Config script for the secondary file in step 24 that associates the button with a JumpID command that specified the contents

topic in the primary file. A similar script in the primary file, created in step 4, associates a Contents() command with that button. These scripts ensure that the button always causes the proper topic to be displayed, because they are executed every time the file is loaded or a window is displayed. Each jump between files thus causes the new file's script to be executed.

**Comment**

As you saw in this How-To, interfile popups *can* be defined with no side effects. Unfortunately, such a design rarely provides any useful value.

Simple interfile jumps from hot spots are also practical, as long as the Contents button is redefined properly and the appearance of the History list is acceptable.

This case was simple because you were able to redefine the Contents button every time either file was displayed. What if you need to change the button based on which topics are displayed or are about to be displayed? This is not as simple as it seems—where are you going to execute the command to redefine the button? If the user can jump into the second file at any topic, you have to be sure the right command is always executed. You also have to be able to change it back when the user returns to the original file. The multifile support in the Back and History buttons can make this a complex problem, because you might not be able to control your user's path between the files. The best way to solve such problems is to carefully avoid them in your application design.

## Use the Go Back Button Between Files?

Complexity: EASY

### Problem

I want to define keywords in one file that display topics in a second file. The user must be able to use the Go Back button normally, even if the previous topic is in the other file.

### Technique

The keywords in the primary file are defined in a topic containing topic-entry commands that jump to the secondary file and handle the return properly.

You create the standard directories, project file and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and create the primary files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP7\_2, and the standard subdirectories—TEXT, SOUNDS, PICTURES, and MOVIES—under CHAP7\_2. Create these four directories for all projects, even if some are not needed.
2. Use Viewer's Project Editor to create a new project file named CHAP7\_2P.MVP in your VIEWERHT\CHAP7\_2 directory. Enter the name of your document file as TEXT\CHAP7\_2P.RTF.
3. Choose Title Options from the Section menu to display the Title Options dialog box. Enter contents in the Contents Topic field. Click on OK.
4. Choose Config from the Section menu. Paste in a CBB command, then edit it. Select `btn\_contents' from the pull-down list in the ButtonID field, and select Contents() in the Command field. Click on OK in each dialog box to return to Project Editor.
5. Save your project file by choosing Save from the File menu.
6. Use Project Editor to start Word and create your document, and create a new topic with context string contents. Be sure to delete the page break that is created before your topic.
7. Use Topic Editor to define a keyword of primary contents, as shown in Figure 7-2.

```
{ewc vwrht2, TsTextButton, "Figure
7i½2"[Macro=JI('viewerht.mvb>SecWin', `fig7_2')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

8. Use Topic Editor to define a Topic Title of Primary contents.
9. Enter text as a heading line:  
This is the Contents topic in the primary file.
10. Insert a few blank lines and create a hot spot with text Jump to second topic that jumps to context string topic\_2p.



11. Create a new topic with context string `topic_2p`, keyword `primary second` and title `Primary second topic`.
12. Enter the following text:  
This is the second topic in the primary file.
13. Create a new topic with context string `topic_3p`, keyword `special`, and title `primary third`.
14. Choose `Footnote` from Word's `Insert` menu, click on `Custom Footnote Mark`, and enter an exclamation point (!) in the field.
15. Word displays the `Footnote` pane at the bottom of the window, with the insertion point following an exclamation point. Enter the following text on one line, as shown in Figure 7-3:  

```
{vfld2305865549202063371}IfThenElse{vfld12232066859008}
({vfld137438953483}IsMark{vfld12232066859008}("forward"),
"{vfld137438953483}DeleteMark{vfld12232066859008}(`forward');
{vfld137438953483}Back(){vfld12232066859008}",
"{vfld137438953483}SaveMark{vfld3473682146420326400}
(`forward'); JI(`chap7_2s.mvb', `topic_1s')")
```

```
{ewc vwrht2, TsTextButton, "Figure
7i½3"[Macro=JI(`viewerht.mvb>SecWin', `fig7_3')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

16. Click on the `Close` button at the top of the `Footnote` pane.
17. Enter the following text in the topic:  
This topic should never be displayed.
18. Save your document file and compile as usual.

Next create the secondary file:

19. Use `Viewer's Project Editor` to create a new project file named `CHAP7_2S.MVP` in your `VIEWERHT\CHAP7_2` directory. Enter the name of your document file as `TEXT\CHAP7_2S.RTF`.
20. Choose `Config` from the `Section` menu. Paste in a `CBB` command, then edit it. Select `'btn_contents'` from the pull-down list in the `ButtonID` field, and select `JumpID()` in the `Command` field. Edit the `JumpID` parameters to read `JumpID(`chap7_2p.mvb', `contents')`. Click on `OK` in each dialog box to return to `Project Editor`.
21. Save your project file.
22. Use `Project Editor` to start Word and create your document, and create a new topic with context string `topic_1s` and title `Secondary`. Be sure to delete the page break that is created before your topic.
23. Enter the text:  
This message comes to you from the secondary file
24. Save your document file and compile as usual.

Next test your application:

25. Choose `Open` from `Project Editor's File` menu to reopen the primary project file, then run `Viewer` to display the primary `MVB` file.
26. Click on the `Index` button, then select the entry `Primary Second`. See

which topic is displayed.

27. Repeat step 26 for the entry Special. See which topic is displayed and examine the History list.
28. Click on the Go Back button. See which topic is displayed, and examine the History list again. It should look like Figure 7-4.

```
{ewc vwrht2, TsTextButton, "Figure  
7ic1/24"[Macro=JI('viewerht.mvb>SecWin', `fig7_4')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

29. Try other operations, including using the Contents button from each file.

### How It Works

The Config scripts for both the primary and secondary files, created in steps 4 and 20, redefine the command associated with the Contents button just as in How-To 7.1. This assures that the button is always associated with a command that causes the proper topic to be displayed, no matter which file is active. (This is explained in detail in How-To 7.1.)

The topic where the keyword *special* is defined is never seen—the topic entry command is executed immediately, and it always causes a jump to a different topic. This topic’s sole purpose is to cause the right topic to be displayed next, based on how you reached this special “traffic-cop” topic. It jumps to the desired topic in the secondary file the first time it is executed, and jumps to the previous topic in the primary file the next time. The jump to the secondary file displays the topic associated with the keyword, and the second jump displays the topic that the user would expect to see after clicking on the Go Back button.

The topic entry IfThenElse command starts by testing for a mark named *forward*. If the mark exists, the commands in its second parameter are executed. If the mark doesn’t exist, the commands in the third parameter are executed. The mark doesn’t exist the first time this topic is displayed, so a SaveMark command is executed, thus creating the forward mark, followed by a JumpID command specifying a topic in the secondary file. The entry command is executed again when you return by clicking on the Go Back button. This time the mark does exist, so a DeleteMark command is executed to remove it, followed by a Back() command to redisplay the preceding topic.

This topic entry command only works properly if the topic is entered in the expected sequence. What would happen if you returned to the primary file by using the History button, then used the Index button and selected the special keyword again? The mark would still exist, so Viewer would delete it and execute the Back() command instead of jumping to the secondary file.

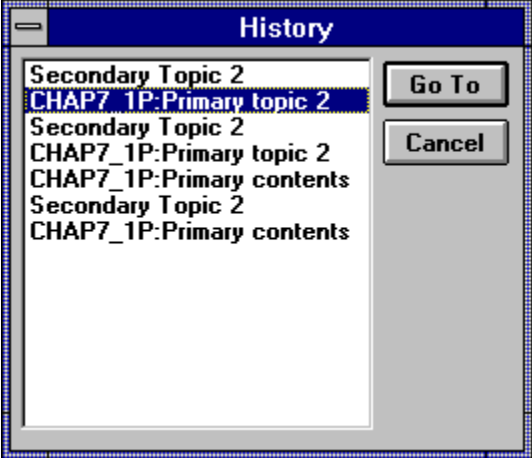
### Comment

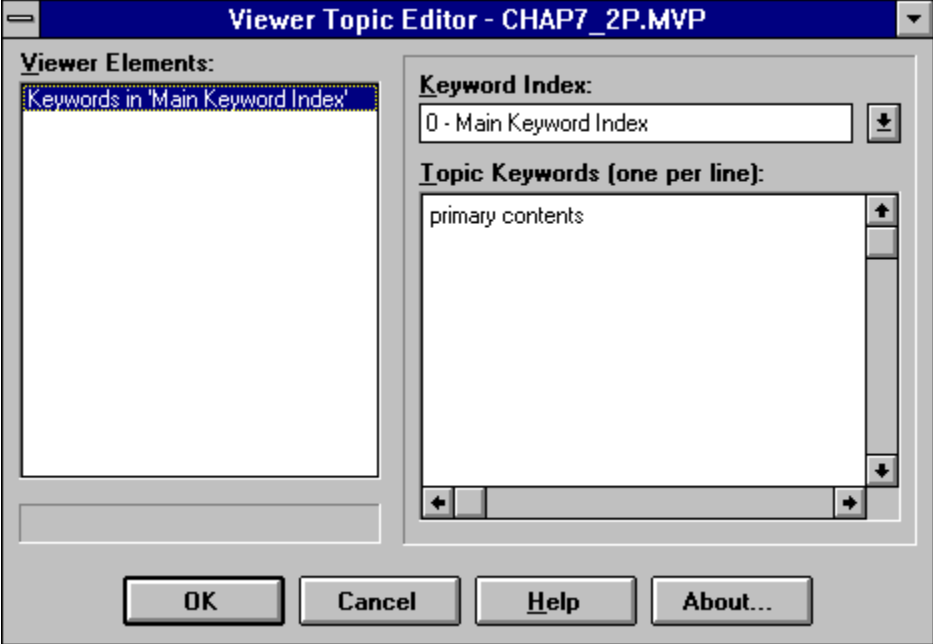
The potential problem this How-To solves has often occurred in the Windows Help system when authors want to make use of existing help files. It should never arise in Viewer except as a result of poor planning. As you can see, there aren’t any good solutions other than avoiding the problem in

the first place.

### 7.3 Tips and Tricks

- ⇒ If you want to use {vfld137438953482}secondary windows {vfld13331578486784} in conjunction with multiple files, you must be careful to define the {vfld137438953484}secondary window {vfld4926370438184960} in the file containing the topics displayed in that window. It doesn't matter which file contains the command or hot spot that causes the topic to be displayed.
- ⇒ If you run into other problems, you might be able to solve them by defining all of the topics in the secondary file as part of a special topic group. This group would exist solely to support group entry or group exit commands similar to the command in How-To 7.2. Note that the Viewer manual describes the sequence of group and topic commands incorrectly. When entering a group, the group commands are executed first, and then the topic commands.
- ⇒ If Viewer reports that it can't find your alternate file, it may be because the current directory isn't the project directory. You can solve this by starting the File Manager, selecting the project directory (which sets the current directory), and double-clicking on the primary MVB file. Viewer searches for files using the standard Windows sequence—the current directory, the Windows directory, the Windows System directory, and the directories listed in the DOS PATH command in the AUTOEXEC.BAT file. If the file still isn't found, Viewer checks the [FILES] section of the VIEWER.INI file. This INI file is described in Appendix A.oiceñ {vfld137438953482} {vfld7306245429312618496}





Microsoft Word - CHAP7\_2P.RTF

File Edit View Insert Format Tools Table Window Help

footnote text Times New Roman 10

#E3- This is the Contents topic in the primary file.

¶

¶

Link to second topic in 2p

---

#E3- This is the second topic in the primary file.

---

#E3- This topic should never be displayed.

---

**Footnotes** Close

#topic 2p

K0.special

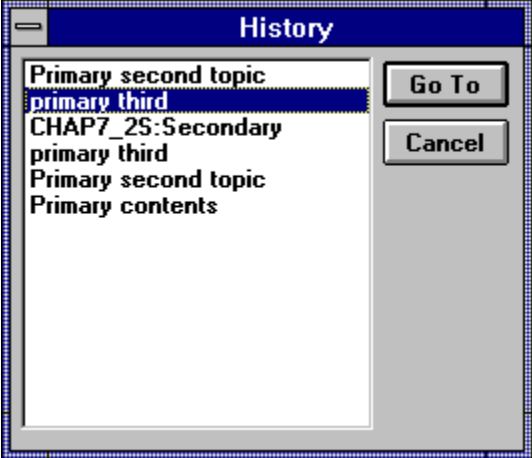
\$primary third

! IfThenElse(IsMark('forward'), "DeleteMark('forward'); Dack()", "SaveMark('forward');

J('chap7\_2s.rvt', 'topic\_1s')

¶

Pg 3 Sec 1 3/ 3 At \_n Col 33 100% NUM







You can make your application more interesting and exciting by using sounds. Viewer gives you a number of techniques that cause a sound to be played. A sound can be played

- ü Automatically, when the application is loaded
- ü Automatically, when a topic is displayed
- ü When the user clicks on a button
- ü When the user clicks on a text hot spot
- ü When the user clicks on a graphic hot spot
- ü When the user clicks on a special control button

Most of these should look familiar to you from previous How-Tos, since these are the same ways that a Viewer command can be executed.

You can play sound files by executing an external Windows function. External commands must be defined to Viewer through Viewer's RegisterRoutine command—then they can be used just like any standard built-in commands.

The special control buttons (the last item in the above list) are produced by a Viewer embedded window command that helps you use both sounds and movies in your application. This command is demonstrated in How-Tos 8.1 and 8.2, and also in Chapter 9. Embedded window commands only function while they are visible in the window. They load the entire sound file into memory as soon as their portion of the topic is displayed. This can require significant time and memory if the file is large.

Two types of sound files are supported by Viewer. {vfld-9223356093936173046} *Wave* files {vfld11132555231232} have a file extension of WAV, and contain a digital representation of the actual sounds. Wave files can contain any sounds desired, such as spoken voices, sound effects, or music. {vfld-9223356093936173046} *MIDI* files {vfld280933810831360} have a file extension of MID, and contain instructions to an electronic synthesizer. These instructions tell the synthesizer to play particular notes with specified instrument sounds, durations, and rhythms. Several instruments can play at the same time, and each can play several notes together. MIDI files can only play music or other sounds built into the synthesizer.

A sound card, external speakers, and associated software drivers must be installed in the computer in order to play sound files. Viewer does not support the speaker built into the computer, or the driver software used with that speaker. How-Tos 8.3 through 8.5 show you how to test for the needed drivers, and alter your application's behavior accordingly. Many companies make sound cards with a wide range of capabilities. Two of the most popular product lines are Pro Audio Spectrum (made by MediaVision) and Sound Blaster (made by Creative Labs).

Wave files can be created using programs that are distributed with sound cards and drivers. For example, the MediaVision Pro Audio Spectrum card comes with a program called Pocket Recorder. Commercial programs are also available to serve users with more exacting requirements. Viewer's WaveEdit program can be used to modify an existing wave file, as described in section 2.4.

MIDI files are created and edited by specialized commercial programs known as *sequencers*. These range from limited-function packages to high-powered systems designed for professional musicians. A popular line of

midrange packages is sold by MidiSoft.

## Play a Sound?

Complexity: EASY

### Problem

I want the user to hear selected sounds by clicking on a button. The sounds could be produced from either wave or MIDI files.

### Technique

You use Viewer's multimedia command to create the standard set of control buttons.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP8\_1, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP8\_1. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the sound files for this How-To from the VIEWERHT\HOWTOS\CHAP8\CHAP8\_1\SOUNDS directory on the CD-ROM to the VIEWERHT\CHAP8\_1\SOUNDS subdirectory on your hard drive.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP8\_1 directory. Enter the name of your document file as TEXT\CHAP8\_1.RTF.
4. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this topic doesn't need a context string. Viewer displays the first topic when a file is loaded unless you specify otherwise.

Now you can proceed with adding a sound:

5. Invoke Topic Editor and select Multimedia (using ewX with MVMCI2). Click on the Options button to display the Multimedia Options dialog box.
6. Select the file SOUNDS\CHAP8\_1.WAV, and click on the Store File in Baggage check box.
7. Leave the MCI Device field as {vfld137438953482} WaveAudio {vfld-9223349496866406400}, leave the Show Controller check box selected, leave the Text-Aligned Position selected, and leave the Play Entire File check box selected. Click on the Layout button and enter the caption Wave file. The completed dialog box should look like Figure 8-1. Click on OK twice to return to the document.

```
{ewc vwrht2, TsTextButton, "Figure  
8½1 "[Macro=JI('viewerht.mvb>SecWin', `fig8_1')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

8. Insert a few blank lines, then repeat steps 5 through 7 to create another controller, selecting MCI Device Sequencer, selecting file SOUNDS\CHAP8\_1.MID, and entering the caption MIDI file. Note that you cannot select the Store File in Baggage check box, because the MCI driver programs that handle these files don't include the necessary support. Click on OK twice to return to your document.
9. Save your file and compile as usual. The window should show the standard multimedia controllers with captions, as shown in Figure 8-2.

```
{ewc vwrht2, TsTextButton, "Figure
8½2"[Macro=JI('viewerht.mvb>SecWin', `fig8_2')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

10. Click on the second button (with the right arrow) on the control labelled Wave File. This is the Play/Pause button. The wave file plays, and the slider moves to show what portion of the file is playing. Click on the first button, containing a square, while the file is playing—the sound stops and the slider returns to the beginning. Drag the slider and click on the Play/Pause button—the file begins playing at the point represented by the slider's position.
11. Repeat step 10 for the MIDI controller. Also click on the Play/Pause button while the file is playing. It pauses, and resumes from the same point when you click on it again.

### How It Works

The command you inserted with Topic Editor displays the standard multimedia controller buttons. The standard controller contains a Play/Pause button, a Stop button, and a progress slider. How-To 8.2 shows you how to design your own controller. You usually should specify a caption to provide a label for the controller.

Each type of file uses a different MCI device. Wave files use the WaveAudio device, and MIDI files use the Sequencer. Other device names are used in Chapter 9.

### Comment

The controller is not seen if the Show Controller check box is cleared. This is only useful if the Auto-Start check box is set—otherwise, how would your file be played?

A Multimedia controller can be placed anywhere—within the body of a topic, in the non-scrolling region, in a separate pane, in a popup window, or in a floating secondary window. Consider the visual impact of each possible location while designing your application.

The purpose of multimedia controllers is not obvious to your users. Be sure to make the purpose of each one clear, and provide instructions for their use.



## Add Custom Control Buttons?

Complexity: EASY

### Problem

I want to give my users more control over sound playing than just the play and stop buttons in the standard controller.

### Technique

You use Viewer's multimedia command to create a custom set of control buttons.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP8\_2, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP8\_2. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the sound files for this How-To from the VIEWERHT\HOWTOS\CHAP8\CHAP8\_2\SOUNDS directory on the CD-ROM to the VIEWERHT\CHAP8\_2\SOUNDS subdirectory on your hard drive.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP8\_2 directory. Enter the name of your document file as TEXT\CHAP8\_2.RTF.
4. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this topic doesn't need a context string. Viewer displays the first topic when a file is loaded unless you specify otherwise.

Now you can create the customized controller:

5. Invoke Topic Editor and select Multimedia (using ewX with MVMCI2). Click on the Options button to display the Multimedia Options dialog box.
6. Select the file SOUNDS\CHAP8\_2.WAV, and click on the Store File in Baggage check box.
7. Leave the MCI Device field as WaveAudio, leave the Show Controller check box selected, leave the Text-Aligned Position selected, and leave the Play Entire File check box selected. Click on the Layout button and enter the caption Custom Controller. Click on OK.
8. Click on the Edit Controller button to display the MCI Controller dialog box.
9. Click on the Custom Controller Type radio button to enable the custom choices.
10. Click on the Play option. Notice that the Play/Pause button option is

unselected when you do this—you can't have both at once. Click on the Pause option.

11. Click on the Scan Backward and Scan Forward options. Notice that the Step Size field is activated and filled in as soon as you select one of these buttons. Leave that field set for 1000 milliseconds (1 second). The completed dialog box should look like Figure 8–3

```
{ewc vwrht2, TsTextButton, "Figure  
8i½3"[Macro=JI('viewerht.mvb>SecWin', `fig8_3')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

12. Leave the Eject, Previous Section, and Next Section options off. Leave the slider options selected, and click on OK in each dialog box until you return to the document.
13. Save the document file, and compile and test as usual. The control should look like Figure 8–4. Try each of the buttons.

```
{ewc vwrht2, TsTextButton, "Figure  
8i½4"[Macro=JI('viewerht.mvb>SecWin', `fig8_4')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

The multimedia command inserted by Topic Editor displays the controller with the buttons you selected. The standard controller contains a Play/Pause button, a Stop button, and a progress slider. This How-To demonstrates all of the custom buttons suitable for sound files: Scan Backward, Scan Forward, Stop, Pause, and Play.

### Comment

The {vfld137438953482}eject button{vfld1113255231232} ejects a CD when playing {vfld137438953482}CDAudio{vfld-7998112004399169536} files. Previous Section and Next Section buttons apply to files that are divided into sections. Sections are demonstrated in How-To 9.3.



**Play Sounds Automatically?**

Complexity: INTERMEDIATE

**Problem**

I want to play a sound file automatically when a topic is displayed.

**Technique**

You use two Windows API external commands. The first command, `{vfld137438953483}waveOutGetNumDevs{vfld12232066859008}`, lets you test for the presence of the wave drivers required. The second, `{vfld137438953483}sndPlaySound{vfld8142789060096688128}`, is used to play the sound file.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

**Steps**

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP8\_3, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP8\_3. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the sound file for this How-To from the VIEWERHT\HOWTOS\CHAP8\CHAP8\_3 directory on the CD-ROM to the VIEWERHT\CHAP8\_3 subdirectory on your hard drive. Note that you are *not* copying it to the SOUNDS subdirectory this time, as you did in most previous How-Tos.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP8\_3 directory. Enter the name of your document file as TEXT\CHAP8\_3.RTF.

Next define the external commands, and then execute them:

4. Choose Config from the Section menu, then click on the Paste Command button and select the RegisterRoutine command. Click on OK.
5. Click on the Edit Command button and enter ``mmsystem'` in the DLLName field, ``waveOutGetNumDevs'` in the FunctionName field, and ``u='` in the Parameters field. The completed dialog box looks like Figure 8-5. Click on OK.

```
{ewc vwrht2, TsTextButton, "Figure
8i½5"[Macro=JI('viewerht.mvb>SecWin', `fig8_5')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

6. Repeat steps 4 and 5 to insert another RegisterRoutine command in the script, entering ``mmsystem.dll'` as the DLLName, ``sndPlaySound'` as the FunctionName, and ``Su'` as the Parameters. Click on OK.

7. The completed script should look like Figure 8–6.

```
{ewc vwrht2, TsTextButton, "Figure  
8½6"[Macro=JI('viewerht.mvb>SecWin', 'fig8_6')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

8. Save the updated project file.

Now create the document file:

9. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this topic doesn't need a context string. Viewer displays the first topic when a file is loaded unless you specify otherwise.
10. Insert a few blank lines, and create a text hot spot with the text Go to next topic, which jumps to context string topic\_2.
11. Create a new topic with context string topic\_2.
12. Use Topic Editor to create a topic entry command, using the IfThen command. Enter waveOutGetNumDevs() in the Condition field, and "sndPlaySound('chap8\_3.wav', 0)" in the Command field.
13. Enter a line of text as follows:

You should have heard the sounds if you have a sound card.

14. Save the document file, and compile and test as usual. When you click on the hot spot you see the second topic. The sound file is played if you have a sound card and drivers installed.

### How It Works

The RegisterRoutine command, in steps 4 through 6, defines external commands. The first parameter is the name of the file containing the command. These files usually have DLL as the file name extension, and are known as DLL files. The second parameter is the name of the command being defined, and the third defines the data types of the command's parameters. For example, *S* is used to indicate a string, and *u* an unsigned integer. If an equal sign is part of this parameter, the single character preceding the sign defines the value returned by the command. The waveOutGetNumDevs command in step 5 returns an unsigned integer. The sndPlaySound command in step 6 takes two parameters—a string and an unsigned integer.

The {vfld137438953483}waveOutGetNumDevs{vfld12232066859008} command returns the number of wave output devices that are present in the computer. This usually is 1 if a sound card and drivers are installed, or 0 if not. A similar command, {vfld137438953483}midiOutGetNumDevs{vfld3131967461654528}, returns the number of MIDI output devices. These commands are part of the standard Windows MMSYSTEM.DLL file that includes most of the Windows multimedia functions.

The `{vfld137438953483}sndPlaySound{vfld11132555231232}` command plays wave files. No user controls are used—it plays the file as soon as it is executed. The first parameter is the name of the sound file or the name of a system sound defined in the [sounds] section of the `{vfld137438953482}WIN.INI{vfld8070731466058760192}` file, such as SystemQuestion. This command does not support Baggage—any files used by this command must be separate from the MVB file. This command also does not support paths, which is why the file had to be copied into the same subdirectory as the MVB file. The second parameter is referred to as the flag parameter. It contains a code, where 0 means that the command should not return to Viewer until the file has finished playing, and 1 means that Viewer continues with other commands or operations while the sound continues to play.

In step 12 you define an `{vfld137438953483}IfThen{vfld72057052872048640}` command that uses the external commands just described. The IfThen command is defined as a topic entry command, which means that it executes when the topic is displayed. The IfThen command treats a value of 0 in its condition as false, and any other value as true. It only executes the command in its second parameter if the condition is true. The return from waveOutGetNumDevs thus corresponds perfectly—0 means that sound is not supported, and 1 means it is supported.

### **Comment**

The RegisterRoutine command lets you extend the capabilities of Viewer to include functions in Windows DLLs or custom-written commands. The ability to use values returned by external commands makes it possible for your Viewer application to alter its operation based on information obtained automatically from the user's computer.

The names of external commands defined through the RegisterRoutine are case-sensitive. In other words, `sndPlaySound` works but `SndPlaySound` doesn't. These errors don't cause error messages—the command just isn't executed. You can go crazy trying to find such mistakes!

The IfThen command cannot be used with embedded commands such as the multimedia control used in How-Tos 8.1 and 8.2.

The `sndPlaySound` command is described in detail in How-To 8.5.

**Add a Spoken Greeting?**

Complexity: INTERMEDIATE

**Problem**

I want to greet users with a spoken message when my application is first loaded.

**Technique**

You use the Windows `waveOutGetNumDevs` command to determine if sounds are supported, and the `sndPlaySound` command to play the file. The commands execute in the Config script.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

**Steps**

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP8\_4, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP8\_4. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the sound files for this How-To from the VIEWERHT\HOWTOS\CHAP8\CHAP8\_4 subdirectory on the CD-ROM to the VIEWERHT\CHAP8\_4 subdirectory on your hard drive. Note that you are *not* copying it to the SOUNDS subdirectory this time, as you did in most previous How-Tos.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP8\_4 directory. Enter the name of your document file as TEXT\CHAP8\_4.RTF.

Next define the external commands, and then execute them:

4. Choose Config from the Section menu, then click on the Paste Command button and select the RegisterRoutine command. Click on OK.
5. Click on the Edit Command button and enter 'mmsystem' in the DLLName field, 'waveOutGetNumDevs' in the FunctionName field, and 'u=' in the Parameters field. Click on OK.
6. Repeat steps 4 and 5 to insert another RegisterRoutine command in the script, entering 'mmsystem.dll' as the DLLName, 'sndPlaySound' as the FunctionName, and 'Su' as the Parameters. Click on OK.
7. Paste in an IfThen command, and enter `waveOutGetNumDevs()` in the Condition field. Change the Command field to `"sndPlaySound('chap8_4.wav', 0)"`
8. The completed dialog box looks like Figure 8-7. Click on OK in each dialog box, then save your project file.

{ewc vwrht2, TsTextButton, "Figure

8i½7"[Macro=JI('viewerht.mvb>SecWin', `fig8\_7')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}

Next create a minimal document file:

9. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this topic doesn't need a context string. Viewer displays the first topic when a file is loaded unless you specify otherwise.
10. Enter a line of text:  
If you have a sound card, you should have heard the greeting.
11. Save the document file, and compile and test as usual.

### How It Works

The external commands are defined through the RegisterRoutine commands in steps 4 through 6. They are executed within the IfThen command that is added to the Config startup script in step 7. These commands are described in How-Tos 8.3 and 8.5.

The sndPlaySound command does not support {vfld137438953484} path names {vfld2377762623132270592}. This is why you had to copy the sound file into the same subdirectory as the MVB file. The sequence of directories searched for the file is described in How-To 8.5.

In this case, the IfThen command is executed when the file is loaded, rather than when a topic is displayed, by placing it in the Config script instead of defining it as a topic entry command. This demonstrates the variety of effects that you can provide for your user by combining common operations in different ways.

The Viewer {vfld137438953484} window {vfld13331578486784} does not {vfld137438953484} appear {vfld-9223356093936173056} until after the sound file has finished because you used 0 in the flag parameter. How-To 8.3 explains that 0 means the processing should not continue until the file has been played, and 1 means that other processing *should* continue. If you change the value to 1 and recompile, the window appears during the greeting.

### Comment

The command to play the sound file is executed as part of the Config script so that it executes when the file is loaded. This also executes every time a secondary window is displayed from this file, or after a jump from a different MVB file. How-To 5.10 demonstrates an alternative method of executing a command only when the application is initially loaded.

A similar {vfld137438953482} good-bye message {vfld-9223356093936173056} could be played by adding the proper commands to the Exit button and the Exit choice from the File menu. Be sure to use 0 as the flag parameter with the sndPlaySound command, so that the Exit() command is not executed until the sound file has finished playing.



**Play Continuous Background Music?**

Complexity: INTERMEDIATE

**Problem**

I want to play background music continuously while the user performs other operations.

**Technique**

This example demonstrates two ways to play a sound file continuously. One uses the standard multimedia control used in How-Tos 8.1 and 8.2. The other uses the `sndPlaySound` external command as in How-Tos 8.3 and 8.4.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

**Steps**

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP8\_5, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP8\_5. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the sound file for this How-To from the VIEWERHT\HOWTOS\CHAP8\CHAP8\_5 subdirectory on the CD-ROM to the VIEWERHT\CHAP8\_5 subdirectory on your hard drive. Note that you are *not* copying it to the SOUNDS subdirectory this time, as you did in most previous How-Tos.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP8\_5 directory. Enter the name of your document file as TEXT\CHAP8\_5.RTF.

Next define the external commands and a topic group:

4. Choose Config from the Section menu, then click on the Paste Command button and select the RegisterRoutine command. Click on OK.
5. Click on the Edit Command button and enter 'mmsystem' in the DLLName field, 'waveOutGetNumDevs' in the FunctionName field, and 'u=' in the Parameters field. Click on OK.
6. Repeat steps 4 and 5 to insert another RegisterRoutine command in the script, entering 'mmsystem.dll' as the DLLName, 'sndPlaySound' as the FunctionName, and 'Su' as the Parameters.
7. Enter the following command in the script, all on one line, then click on OK:

```
IfThen(waveOutGetNumDevs(), `SaveMark("soundOK"))
```

8. Choose Groups from the Section menu, then click on the New button to define a group with the default name of Group1. Clear the Searchable check box, then click on the Exit Script button. Enter the following command in the script, all on one line:

```
IfThen(IsMark("sndPlaySound"), `sndPlaySound("chap8_5.wav", 2);
```

DeleteMark("sndPlaySound")')

9. The completed dialog box looks like Figure 8–8. Click on OK twice to return to the main Project Editor window, then save the project file.

```
{ewc vwrht2, TsTextButton, "Figure  
8i½8"[Macro=JI('viewerht.mvb>SecWin', `fig8_8')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Now you can create the document file:

10. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this topic doesn't need a context string. Viewer displays the first topic when a file is loaded unless you specify otherwise. Enter the following text:

This is the Contents topic.

11. Insert a few blank lines and create a hot spot with text Jump to the second topic that jumps to context string topic\_2.
12. Create a new topic with context string topic\_2.
13. Use Topic Editor to add a Topic Groups footnote. Leave the Topic Browse Sequence and Browse Sequence Number fields blank. Enter Group1 in the Topic Groups field and click on OK.

14. Enter the following text:

This is the second topic. Click on one of the following controls to hear some music while you scroll through the topic.

15. Insert a couple of blank lines, and create a hot spot with text Play music using sndPlaySound. The hot spot should execute the following command:

```
IfThen(IsMark("soundOK"), "sndPlaySound(`chap8_5.wav',  
9);SaveMark(`sndPlaySound')
```

16. Insert another couple of blank lines, and use Topic Editor to add a Multimedia command. Select the Sequencer MCI Device and file name SOUNDS\CHAP8\_5.MID, and select the Looping check box. Enter Multimedia Controller as the caption. Click on OK twice to return to the document.

17. Insert a couple of blank lines, then enter 30 lines of text saying This is line 1, This is line 2, through This is line 30.

18. Insert a blank line and enter a line of boldfaced text as follows:

Now use the Contents or Go Back button to return to the Contents topic.

19. Save your document file, and compile and test as usual.

20. Click on the hot spot in the Contents topic. The second topic should look like Figure 8–9.



```
{ewc vwrht2, TsTextButton, "Figure  
8i½9"[Macro=JI('viewerht.mvb>SecWin', 'fig8_9')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

21. Click on the hot spot, then scroll through the topic. Click on the Go Back button to return to the Contents topic.
22. Repeat steps 19 and 20, using the multimedia controller instead of the hot spot in the second topic. The sound should stop when the controller scrolls off the window.

### How It Works

You define the external commands that determine if the wave file drivers are installed, and that play wave files, in steps 4 through 6. How-To 8.3 explains these definitions in detail. In step 7 you use an IfThen command in the startup Config script to create a mark named soundOK if the drivers are installed. In step 8 you define the topic group that is explained later in this section.

You create two topics. The Contents topic exists just so you can leave the second topic and hear the sound stop without exiting Viewer. The hot spot in the Contents topic lets you jump to the second topic, where you really do everything!

The hot spot you create in step 15 first checks for the soundOK mark. If that's missing, the hot spot doesn't do anything. In a real application you might display a popup here that says *Sorry, you can't do that*, or just not display this topic. If the soundOK mark is present, another mark named sndPlaySound is created and the sound is played. This new mark indicates that you've started the sound, and it needs to be stopped. It's used shortly. The flag parameter in the command playing the sound is a new value (9) that you haven't used before. This means keep playing the file until you say stop, and return to Viewer immediately. The codes for these functions are 8 and 1 respectively. You indicate multiple options by adding the values together (9 is used to represent 8 and 1). The complete set of flag values you can use are defined in the Comment section below.

The sound stops when you return to the Contents topic, because you defined a topic group exit script in step 8, and the second topic is the only one in that group. The exit script checks for the sndPlaySound mark that indicates you started the sound. If the mark exists, the same sound is played without repetition. This cancels the previous playing. The mark is then deleted to indicate that a shut-off is no longer needed.

When you try the multimedia controller, the sound should continue until the controller scrolls off the window. All of the Viewer ewX commands execute only while their portion of the topic is visible. This technique would work without a problem if the controller were located in a non-scrolling region or a regular pane.

### Comment

Did you try exiting Viewer while the sound is being played by the sndPlaySound command? If not, try it and see what happens. The topic group

exit script does not execute when leaving Viewer, so the sound doesn't stop. It actually keeps playing even after Viewer has ended! You can stop it by playing another wave file, perhaps by using the Windows MPlayer program or by reexecuting the Viewer application you just ended. The best way to prevent this in a real application is to modify the Exit button and the Exit menu item to include commands similar to those you used in the topic group exit script. If you don't do this, you could *really* surprise your users!

The {vfld137438953483}sndPlaySound{vfld1113255231232} command {vfld137438953482}parameters{vfld-9114861779770408960} are defined as follows:

The sound name may be an entry in the [Sounds] section of your WIN.INI file or the name of a file. The current directory is searched first, followed by the Windows directory, the Windows system directory, the directories listed in the DOS path environment variable, and the directories mapped in a network. If no match is found, the SystemDefault sound defined in the WIN.INI file is played. If there is no default entry or its sound file can't be found the function makes no sound.

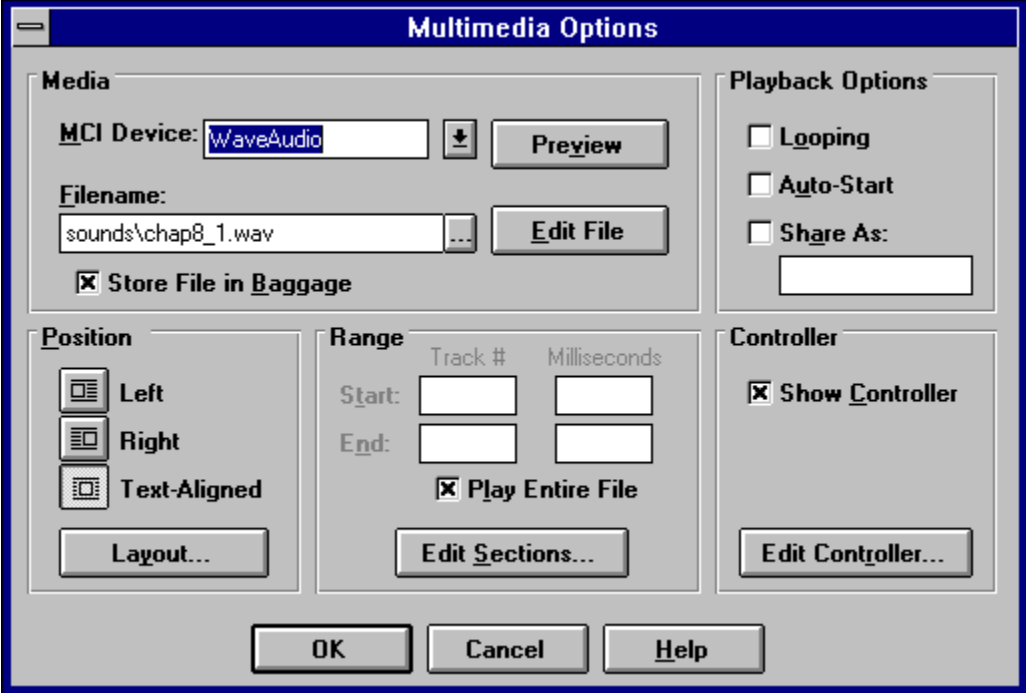
The flag values are as follows (remember, use the sum of the values you want):

- 0—Function does not return until the sound ends.
- 1—Function returns immediately. Terminate the sound by playing the same sound or another sound with the flag set to 0.
- 2—If sound file can't be found, return silently without playing the default sound.
- 8—Sound repeats until ended. You must also specify the 1 flag.
- 16—If a previous sound is currently playing, return immediately without playing the new sound. Returns value of False.

The command returns True if the sound (or the default) is played, otherwise False.

## 8.6 Tips and Tricks

- ⇒ MIDI files created in the Windows 3.0 file format cause Windows 3.1 to issue a confusing warning message when they are played. The message says *This file may not play correctly with the {vfld137438953484}default MIDI setup{vfld-9079242005371944960}*. The message box includes a check box used to indicate that this warning should not be shown thereafter. Such files can be converted to the Windows 3.1 format by the markmidi program included with Microsoft's Visual C++ package and its Software Development Kit (SDK).
- ⇒ Use only MIDI files that are based on the {vfld137438953482}General MIDI {vfld-9223356093936173056} standard included in Microsoft's *Authoring Guidelines for MIDI Files*. This standard specifies which sound numbers represent which instrument sounds—for example, sound number 3 is a honky-tonk piano. You should also be sure that your sound card follows that standard. The popular cards do follow the standard, but some made for professional musicians do not. If you play a MIDI file that was written under one standard on a sound card that uses a different standard, the results are likely to be painful because the wrong instruments are used. Just imagine Brahms' *Lullaby* played with cymbals and a tuba! Also remember that your users could have cards that don't follow the standard.
- ⇒ Wave files can easily be very large. The portions of section 2.2 that pertain to sound files should be studied carefully to avoid creating files that are larger than necessary.
- ⇒ The Wave File Editor and File Conversion programs described in section 2.4 can be very useful in preparing sound files.
- ⇒ For an excellent example of the use of sounds, examine Microsoft's Multimedia Beethoven.
- ⇒ Movie files, described in Chapter 9, can include sounds similar to wave files within the movie. Animation files can cause wave or MIDI files to be played while the animation is playing.









**MCI Controller**


**Controller Type**



Viewer Standard       Custom



**Button Options**

 Play        Stop

 Pause        Eject


 Play/Pause

 Previous Section        Next Section

 Scan Backward        Scan Forward


Step Size:  Milliseconds

**Menu Options**

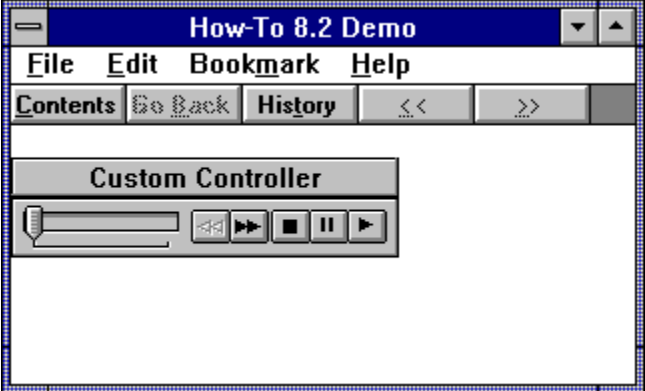
 Include Pop-Up Menu

Show Mute Menu Item

**Slider Options**

Show Slider 

Allow User to Drag Slider



**Edit Command**

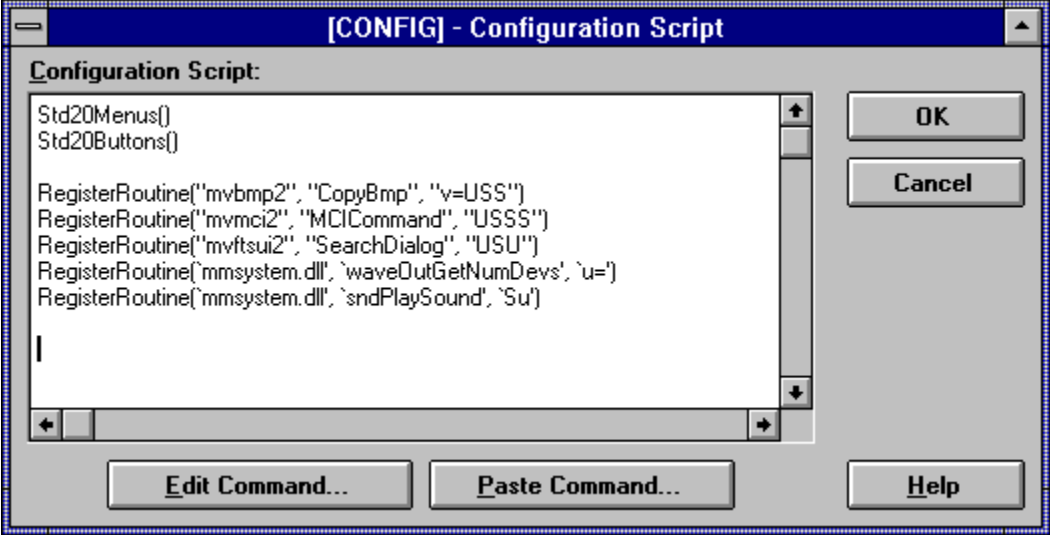
**Edit Command:** RegisterRoutine

**DLLName:**

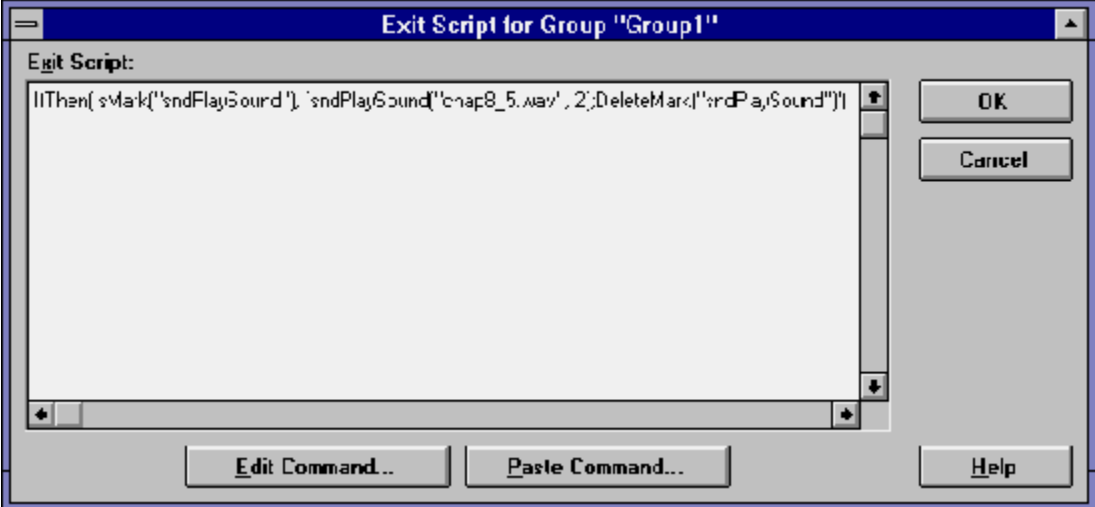
**FunctionName:**

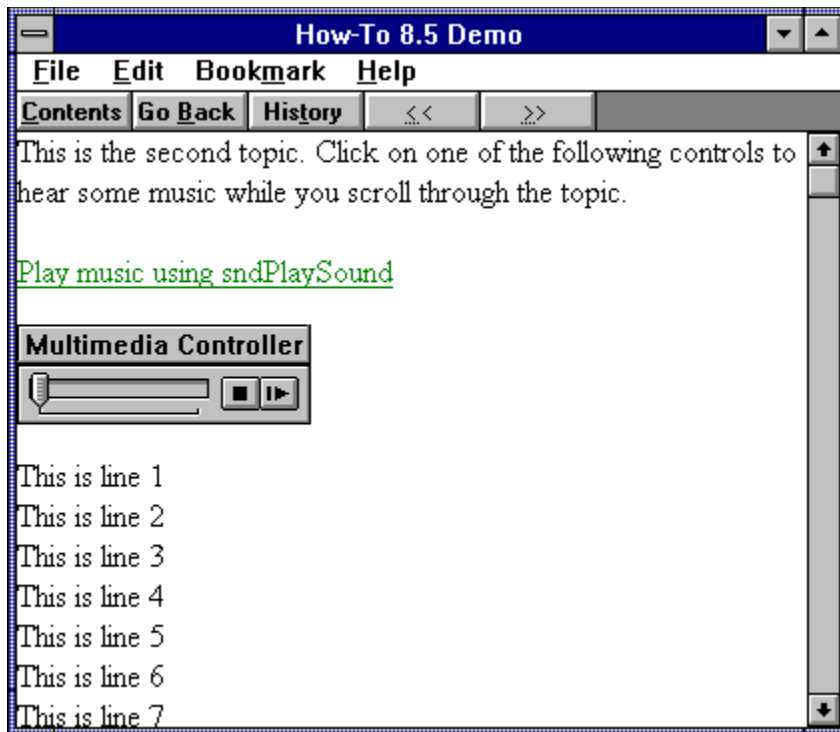
**ParameterSpec:**













Movies and animations can add tremendous value and excitement to your application. The ability to play a recorded movie with reasonable fidelity is a recent development. One of the earliest and best-known applications using movies on a PC is National Geographic's *Mammals*. The use of movies made that package an immediate hit. Animations have been used for several years, with rapidly increasing quality readily available to the PC user.

The term *movies* is used to describe moving pictures that were originally recordings of a live subject. They are usually created with a video camera and converted to digital form with the aid of special PC boards and software. Special hardware is not required to play these files, as long as the video system is adequate. Movies may require displays that support 256 colors. Movies routinely include recorded sounds that are coordinated with the picture. Even a fairly small movie picture area requires displaying a very large number of bits every second. Programmers continue to find ways of reducing the number of bits that must be redrawn, but this is still a new and rapidly advancing technology. Movie files are very large—the shortest are usually more than 1 megabyte, and many are 8 to 10 megabytes long. The movie files used by Viewer have a file extension of `{vflid137438953482}AVI{vflid11132555231232}`, and are based on the Microsoft Video for Windows standard. Viewer is also capable of playing some movie files created using software by Macromedia on a Macintosh computer. These files have an extension of `{vflid137438953482}MMM{vflid2965619813457592320}` and need a special program to convert them to an acceptable format. It can be difficult to get the right file format and drivers to support these files.

The term *animations* is used to describe moving pictures that are created on a PC, using software by vendors such as Gold Disk and Autodesk. No special hardware is required to create or play these files, as long as the video system is adequate. Animations may require displays that support 256 colors. Animations can be as simple as basic cartoons, or as complex as the computer-generated images included in recent Hollywood movies. Animation files can be much smaller than movies, because the animation files only need to contain instructions describing the objects and their movements. Animation files may also produce larger picture areas or better resolution than movies. These files commonly include commands that cause the driver software to play sound files while the pictures are displayed.

Viewer can display any movie or animation files if suitable Windows `{vflid137438953482}MCI{vflid71919613918576640}` drivers are installed. These drivers must be distributed and installed with your application for the files to be displayed properly. This chapter demonstrates the techniques for playing three types of files:

- ü Microsoft AVI movies, which can be selected as a fully supported option within the Viewer Topic Editor Multimedia Options dialog box
- ü Animations with MCI drivers compatible with Viewer; the Multimedia Options dialog box allows these file types to be entered, and the files played properly
- ü Animations without compatible MCI drivers; although these files cannot be played by the standard Viewer multimedia controller, they can be played by other methods

Movies and animations can be played automatically or as a result of user actions, under the same conditions and using similar methods as for sounds. The multimedia controller used in Chapter 8 to play sound files is also used to play movie or animation files that have compatible MCI drivers. Other techniques can be used to play some animation files that do not have compatible MCI drivers.

## Add a Movie?

Complexity: EASY

### Problem

I want to allow the user to play a movie within a topic. I want text to wrap around the movie area just like it wraps around a picture.

### Technique

You insert a standard Viewer multimedia controller into your document file using Topic Editor.

You create the standard directories, project file, and document file for this How To. (To review those procedures, refer to sections 3.1 and 3.2.) Note: Movie files are quite large—over 1 megabyte each. You may need to remove the files and directories from previous How-Tos if you don't have enough disk space available.

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP9\_1, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP9\_1. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the movie files for this How To from the VIEWERHT\HOWTOS\CHAP9\CHAP9\_1\MOVIES directory on the CD-ROM to the VIEWERHT\CHAP9\_1\MOVIES subdirectory on your hard drive.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP9\_1 directory. Enter the name of your document file as TEXT\CHAP9\_1.RTF and save the project file.

Now you can create the document file:

4. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this demonstration doesn't require a context string. Viewer displays the first topic when a file is loaded unless you specify otherwise.
5. Enter a couple of lines of text at the top of the topic, as follows:  
This topic demonstrates the appearance and operation of a standard Viewer multimedia controller playing an AVI movie file.
6. Insert a couple of blank lines, then activate Topic Editor and select Multimedia (using ewX with MVMCI2). Click on the Options button to display the Multimedia Options dialog box.
7. Select MCI Device  
{vfld137438953482} AVIVideo {vfld71919613918576640}, select Filename MOVIES\CHAP9\_1.AVI, and select Position Left. Leave the Store File in Baggage check box unselected. The dialog box should look like Figure 9-1 after making your changes.



```
{ewc vwrht2, TsTextButton, "Figure  
9i½1"[Macro=JI('viewerht.mvb>SecWin', `fig9_1')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

8. Click on the Layout button to display the MCI Layout dialog box. Enter President Roosevelt as the Caption Text. Figure 9–2 shows the result. Click on OK in each dialog box to return to the document.

```
{ewc vwrht2, TsTextButton, "Figure  
9i½2"[Macro=JI('viewerht.mvb>SecWin', `fig9_2')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

9. Enter the following text immediately after the command:  
Text is being placed both before and after the controller to show the effect. The controller is left-justified at the beginning of this paragraph.

10. Save the document file, then compile and test as usual. Figure 9–3 shows the window while the movie is playing.

```
{ewc vwrht2, TsTextButton, "Figure  
9i½3"[Macro=JI('viewerht.mvb>SecWin', `fig9_3')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

11. The second button (with the arrow) is the Play/Pause button. The movie begins playing if you click on this button. Play pauses if you click on it again while the movie is playing. It resumes when you click on the same button again. The first button (with the box) is the Stop button. If you click on it while the movie is playing, play stops and the slider returns to the far left. If you click on the Play/Pause button again the movie plays from the beginning. The slider shows the current relative position within the movie while it is playing or paused. It can be dragged to any position while the movie is paused. Play continues from the point indicated by the slider when you click on the Play/Pause button again.

### How It Works

This How–To uses an embedded window command identical to the one used to play sounds in Chapter 8. It is also very similar to the corresponding command used to display pictures in Chapter 4.

The entire operation is performed by the multimedia controller that you created in steps 6 through 8. The most important parts of the operation are selecting the MCI Device and selecting the file to be played. After you select the MCI Device, only files with compatible extensions are displayed.

The Left Position that was selected has the same effect as the corresponding command when inserting a picture. The text is wrapped around the picture area without any margin. This is described in detail in

How-To 4.1.

**Comment**

Movie files should usually be left out of Baggage to reduce overhead, thus preventing problems during playback.

The size of the picture display area is determined from the movie file. The file is opened, and the picture size is measured, as soon as the portion of the topic containing the embedded command is displayed. This ensures that an area of the right size is reserved, and the text and pictures in the rest of the topic are positioned accordingly, before you see the topic. If everything had to be adjusted when you used the Play button, the resulting effect would be very distracting!

## Control the Movie with Custom Buttons?

Complexity: EASY

### Problem

I want to let the user play a movie with greater control than the standard controller provides.

### Technique

You insert a Viewer multimedia controller that includes all of the appropriate buttons.

You create the standard directories, project file, and document file for this How To. (To review those procedures, refer to sections 3.1 and 3.2.) Note: movie files are quite large—over 1 megabyte each. You may need to remove the files and directories from previous How-Tos if you don't have enough disk space available.

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP9\_2, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP9\_2. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the movie files for this How To from the VIEWERHT\HOWTOS\CHAP9\CHAP9\_2\MOVIES directory on the CD-ROM to the VIEWERHT\CHAP9\_2\MOVIES subdirectory on your hard drive.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP9\_2 directory. Enter the name of your document file as TEXT\CHAP9\_2.RTF and save the project file.

Now you can create the document file:

4. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this demonstration doesn't require a context string. Viewer displays the first topic when a file is loaded unless you specify otherwise.
5. Activate Topic Editor and select Multimedia (using ewX with MVMCI2). Click on the Options button to display the Multimedia Options dialog box.
6. Select MCI Device AVIVideo and select Filename MOVIES\CHAP9\_2.AVI. Leave the Position unchanged, and leave the Store File in Baggage check box unselected.
7. Click on the Layout button and enter President Kennedy as the Caption Text. Click on OK.
8. Click on the Edit Controller button to display the MCI Controller dialog box shown in Figure 9-4. Select the Custom Controller Type check box to enable the other options.
9. Click on the Play check box. Notice that the Play/Pause option is

automatically deselected—you can't have both at once.

10. Select the Pause, Scan Backward, and Scan Forward check boxes. Notice that Step Size is automatically activated and a value of 1000 milliseconds (1 second) inserted when you select the first Scan option. Change this value to 250 milliseconds so that you can scan in  $\frac{1}{4}$  second increments. The resulting dialog box should look like Figure 9–4. Click on OK in each dialog box until you return to the document.

```
{ewc vwrht2, TsTextButton, "Figure  
9i½4"[Macro=JI('viewerht.mvb>SecWin', `fig9_4')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

11. Save the document file, and compile and test as usual.
12. The window should look like Figure 9–5. Try each button, drag the slider, and see what happens.

```
{ewc vwrht2, TsTextButton, "Figure  
9i½5"[Macro=JI('viewerht.mvb>SecWin', `fig9_5')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

This How–To uses the standard Viewer multimedia controller, with all of the appropriate optional buttons activated. Most of the choices you skipped are used when a file is divided into sections, as in How–To 9.3.

Throughout this chapter you continue to leave the Store File in Baggage check box unselected to reduce the disk space required. This is explained in How–To 9.1.

### Comment

The movie area and controller can be positioned in the same way as a picture, as shown in How–To 9.1. These options, and the options related to the controller caption, were skipped to simplify the instructions. You should have no difficulty using these features on your own.

## Play a Movie in {vfld137438953482}Sections{vfld3800470531342336}?

Complexity: INTERMEDIATE

### Problem

I have a movie file that contains several sections. I want the user to be able to play the entire file or individual sections as desired.

### Technique

You use the Viewer multimedia controller from the previous How-Tos with additional user controls, and add section definitions.

You create the standard directories, project file, and document file for this How To. (To review those procedures, refer to sections 3.1 and 3.2.) Note: Movie files are quite large—over 1 megabyte each. You may need to remove the files and directories from previous How-Tos if you don't have enough disk space available.

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP9\_3, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP9\_3. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the movie files for this How To from the VIEWERHT\HOWTOS\CHAP9\CHAP9\_3\MOVIES directory on the CD-ROM to the VIEWERHT\CHAP9\_3\MOVIES subdirectory on your hard drive.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP9\_3 directory. Enter the name of your document file as TEXT\CHAP9\_3.RTF and save the project file.

Now you can create the document file:

4. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this demonstration doesn't require a context string. Viewer displays the first topic when a file is loaded unless you specify otherwise.
5. Activate Topic Editor and select Multimedia (using ewX with MVMCI2). Click on the Options button to display the Multimedia Options dialog box.
6. Select MCI Device AVIVideo and select Filename MOVIES\CHAP9\_3.AVI. Leave the Position unchanged, and leave the Store File in Baggage check box unselected.
7. Click on the Layout button and enter Martin Luther King as the Caption Text. Click on OK.
8. Click on the Edit Sections button to display the MCI Sections dialog box.
9. Leave the Defined Within the Embedded Pane radio button at the top of the dialog box selected.

10. Click on the New button to define a new movie section. Enter My Slogan as the Section Name, and leave 0 in the Section Begins at Milliseconds field. Leave the Pause at Beginning of Section and Display Tick Mark on Slider check boxes checked.
11. Repeat step 10, entering The Promised Land in the Section Name field and 25000 in the Milliseconds field. The resulting dialog box should look like Figure 9–6. Click on OK.

```
{ewc vwrht2, TsTextButton, "Figure
9i½6"[Macro=JI('viewerht.mvb>SecWin', `fig9_6')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

12. Click on the Edit Controller button to display the MCI Controller dialog box. Select the Custom Controller Type check box to enable the other options.
13. Click on the Play check box. Notice that the Play/Pause option is automatically deselected—you can't have both at once.
14. Select the Pause, Scan Backward, Scan Forward, Previous Section, and Next Section check boxes. Notice that Step Size is automatically activated and a value of 1000 milliseconds (1 second) inserted when you select the first Scan option. Change this value to 250 milliseconds so that you can scan in ¼-second increments.
15. Select the Include {vfld137438953482} Pop-Up Menu {vfld-8970462913399619584} check box under the Menu Options section, then click on OK in each dialog box until you return to the document.
16. Save the document file, and compile and test as usual.
17. Point to the Pop-Up Menu icon at the left side of the control, then hold down the left mouse button. This is not a “sticky” menu—you must click and drag to choose an item. The window should look like Figure 9–7. Try each button and menu entry and see what happens.

```
{ewc vwrht2, TsTextButton, "Figure
9i½7"[Macro=JI('viewerht.mvb>SecWin', `fig9_7')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

The most important part of this How-To is defining the movie sections in steps 8 through 11. The Section Name entries are used to build the popup menu, and the time entries determine the beginning of each section. The default section names could have been used if the menu was not requested. The check box labeled Display Tick Mark on Slider caused a mark to be placed below the slider bar at the beginning of each section.

Two new buttons were requested—Previous Section and Next Section. These buttons provide an alternative means, besides the menu, for switching between sections. The menu, or the buttons, or both, must be provided. Otherwise your section definitions are meaningless because the user cannot

switch between sections.

The menu disappears as soon as the mouse button is released. Options must be selected by dragging the mouse pointer to the desired entry before releasing.

The Pause menu entry is a toggled value. If it is selected, play automatically pauses at the end of each section.

### **Comment**

You can store section definitions in a separate text file instead of defining them within the embedded command. The separate file is better if you have many sections, if you want to change the section definitions, or if you create the definitions before the topic is created. This section–definitions file is usually stored in Baggage for the same reasons as other files. The format of this file is described in the Viewer documentation.

To get the beginning times of the sections, if you cannot control them when the movie file is created, run the `{vfld137438953482}MPlayer{vfld280933810831360}` utility and load the file to be played. Begin playing the file, then pause when the desired scene appears. You can use the small left and right arrows that appear to the right of the slider to move in small increments to find the desired spot. Note the time shown in the lower–right corner of the window. The time displayed in seconds must be multiplied by 1,000 to convert to the milliseconds required by Topic Editor.

**Play a Movie Automatically?**

Complexity: INTERMEDIATE

**Problem**

I want to display a movie automatically when a topic is displayed.

**Technique**

Two different techniques are demonstrated. One uses the same multimedia controller as in the previous How-Tos in this chapter. This technique lets you decide if the user can control the movie through the buttons demonstrated in previous How-Tos. The other technique executes a separate program through a topic entry command to play the specified movie. This program provides buttons and a slider that are extremely similar to the standard multimedia controller demonstrated in How-To 9.1.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.) Note: Movie files are quite large—over 1 megabyte each. You may need to remove the files and directories from previous How-Tos if you don't have enough disk space available.

**Steps**

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP9\_4, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP9\_4. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the movie files for this How To from the VIEWERHT\HOWTOS\CHAP9\CHAP9\_4\MOVIES directory on the CD-ROM to the VIEWERHT\CHAP9\_4\MOVIES subdirectory on your hard drive.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP9\_4 directory. Enter the name of your document file as TEXT\CHAP9\_4.RTF and save the project file.
4. Use Project Editor to start Word and create your document, and create a new topic with context string topic\_1. Be sure to delete the page break that is created before your topic.

Now create the topic entry command:

5. Use Topic Editor to define a topic entry command, and click on the Paste Command button. Select the `{vfld137438953483}ExecProgram{vfld1477325272146509824}` command and click on OK.
6. Click on the Edit Command button, then enter ``{vfld137438953483}MPlayer{vfld-9223357193447800832} /play /close movies\chap9_4.avi'` in the CommandLine field, and select 0 — Normal from the pull-down list for the ProgramState field. Click on OK twice to return to the document, then press **[CTRL]-**



**[SPACE]** to restore your normal {vfld137438953484}character format {vfld-9223348397354778624}. Topic Editor does not restore the normal format after this command. If you don't press **[CTRL]–[SPACE]** your following text looks like footnote codes!

7. Insert a few blank lines, and insert a hot spot with the text Go to second topic and context string topic\_2.

Next create the embedded multimedia–controller command:

8. Create a new topic with context string topic\_2. Press **[CTRL]–[SPACE]** to restore your normal character format again, as in step 6.
9. Activate Topic Editor and select Multimedia (using ewX with MVMCI2). Click on the Options button to display the Multimedia Options dialog box.
10. Select MCI Device AVIVideo and select Filename MOVIES\CHAP9\_4.AVI. Select the {vfld137438953482}Auto–Start {vfld280933810831360} check box. Leave the Position unchanged, and leave the Store File in Baggage check box unselected. Click on the Show Controller check box to deselect it.
11. Click on the Layout button and enter Merging Traffic as the Caption Text. Click on OK in each dialog box to return to the document.
12. Insert a couple of blank lines and insert a hot spot with the text Go to first topic and context string topic\_1.
13. Save the document file, and compile and test as usual.
14. Use the hot spots to jump between the two topics. Figure 9–8 shows the MPlayer window in the first topic.

```
{ewc vwrht2, TsTextButton, "Figure  
9i½8"[Macro=JI('viewerht.mvb>SecWin', `fig9_8')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

15. Figure 9–9 shows how the movie is displayed in the second topic.

```
{ewc vwrht2, TsTextButton, "Figure  
9i½9"[Macro=JI('viewerht.mvb>SecWin', `fig9_9')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

16. Compare the effects of the two techniques. While viewing the second topic, move the mouse pointer onto the picture. It changes to a pointing finger, just as it does over a hot spot. Click on the picture and see what happens.

### How It Works

The ExecProgram command you define as a topic entry command in steps 5 and 6 runs the Microsoft Media Player (MPlayer) program shown in Figure

9–8. The /play and /close options cause the program to begin playing the specified movie automatically, and to exit the program when the movie is finished. A double {vfld137438953482}backslash{vfld13331578486784} (\\) is needed in the file name because of RTF–file conventions. A single backslash begins certain RTF control codes, and a double {vfld137438953484}backslash{vfld8142789060096688128} shows that you want a backslash character in your text.

The multimedia controller in this How–To is very similar to those in previous sections. This time, you select the Auto–Start check box, which begins playing the movie immediately. You also deselect the Show Controller check box, which eliminates the control buttons and slider. The result is shown in Figure 9–9. Other features, such as custom buttons and sections, could also be used if desired. This requires leaving the Show Controller option checked.

When the controller is not shown, the picture image serves as a Play/Pause button. Clicking on the picture while the movie is playing causes it to pause, and clicking on it again causes it to resume.

### **Comment**

The ExecProgram command returns control to Viewer immediately, which lets the topic be displayed while the movie is being played. The delay before the hot spot appears is the time Windows needs to load the MPlayer program. This technique takes longer to begin playing than the multimedia controller because of the additional overhead of executing a separate program.

The ExecProgram technique can be used anywhere that a Viewer command can be executed—in a hot spot, a button, or topic entry. The embedded command can only be used within a topic. How–To 9.5 demonstrates using a hot spot to display a topic containing an embedded command.

If you use a hidden controller, make sure your users will know that they can click on the picture image. This isn't obvious!

## Play a Movie in a Popup Window?

Complexity: INTERMEDIATE

### Problem

I want to play a movie in a popup window when the user clicks on a hot spot.

### Technique

You do this two different ways. The first creates a standard popup hot spot that displays a topic containing an embedded multimedia controller like the one used in How-To 9.4. The second executes a series of external MCI commands directly within the hot spot.

You create the standard directories, project file, and document file for this How To. (To review those procedures, refer to sections 3.1 and 3.2.) Note: Movie files are quite large—over 1 megabyte each. You may need to remove the files and directories from previous How-Tos if you don't have enough disk space available.

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP9\_5, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP9\_5. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the movie files for this How To from the VIEWERHT\HOWTOS\CHAP9\CHAP9\_5\MOVIES directory on the CD-ROM to the VIEWERHT\CHAP9\_5\MOVIES subdirectory on your hard drive.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP9\_5 directory. Enter the name of your document file as TEXT\CHAP9\_5.RTF.

Next define the external MCI command:

4. Choose Config from the Section menu and paste a `{vfld137438953483}RegisterRoutine{vfld1477325272146509824}` command into the startup script.
5. Edit the RegisterRoutine command and enter ``mmsystem'` in the DLLName field, ``{vfld137438953483}mciSendString{vfld-9223357193447800832}'` in the FunctionName field, and ``U=SSuu'` in the Parameters field. Click on OK in each dialog box, then save the project file.

Now you can create the document file:

6. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this demonstration doesn't require a context string. Viewer displays the first topic when a file is loaded unless you specify otherwise.
7. Insert a few blank lines, and enter the text Pop up a movie.

8. Select the text you just entered and activate Topic Editor. Click on OK to accept the Hot Spot (text) entry that is selected.
9. Click on the Popup radio button under Hot Spot Type.
10. Select the Jump to... element, then enter the `_movie` as the context string. Click on OK.
11. Insert a couple of blank lines, then enter the text `Pop up a movie differently`. Select that text and create another popup hot spot as you did in steps 8 and 9.
12. Click on the Hidden Text is Command(s) radio button, then type in the following commands exactly as shown. Note the single space between the second pair of quotation marks. Enter each command on a separate line.
 

```
{vfld2305865549202063371}mciSendString{vfld2964212438574039040}("open movies\chap9_5.avi alias movie", " ",0,0)
mciSendString("play movie wait", " ",0,0)
mciSendString("close movie", " ",0,0)
```
13. Insert a blank line, then create a new topic with context string `the_movie`.
14. Use Topic Editor to insert a Multimedia command, and click on the Options button.
15. Select MCI Device AVIVideo, and file name `MOVIES\CHAP9_5.AVI`. Select the Auto-Start check box, and deselect the Show Controller check box.
16. Click on the Layout button, and enter `Motorcycle` as the Caption Text. Click on OK in each dialog box to return to the document.
17. Save the document file, and compile and test as usual.
18. The first popup window should look like Figure 9-10.

```
{ewc vwrht2, TsTextButton, "Figure
9i½10"[Macro=JI('viewerht.mvb>SecWin', `fig9_10')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

19. Drag the left edge of your Viewer window to the right before clicking on the second hot spot. The result should look like Figure 9-11.

```
{ewc vwrht2, TsTextButton, "Figure
9i½11"[Macro=JI('viewerht.mvb>SecWin', `fig9_11')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

The first hot spot is a standard popup hot spot, created in steps 7 through 10, that displays a topic containing a multimedia controller. That topic and controller are defined in steps 13 through 16. The controller starts playing automatically, and doesn't have any control buttons.

The second hot spot is also defined as a popup so that its appearance is consistent with the resulting action. It would work the same way even if it were formatted as a jump. The important part of this definition is that this hot spot executes three MCI commands. The first command opens the movie file and assigns an alias (*movie*) to be used in the following commands. The

second command plays the movie file and waits until it is done before returning to Viewer to execute the next command. The last command closes the file. The second parameter for the `mciSendString` command is always a single space enclosed in quotation marks (" "). The third and fourth parameters are always zero. These parameters provide options that are valuable when the command is called from a C or Visual Basic program, but cannot be used when it is called from a Viewer application. The MCI command is an external command, so it is defined in a `RegisterRoutine` command in steps 4 and 5.

The wait option with the MCI play command is critical. If it is missing, the third command executes immediately, closing the file before it even begins to play. You probably noticed that you couldn't do anything until the file finished playing.

### **Comment**

A double backslash (\\) is needed in the file name because of RTF-file conventions. A single backslash begins certain RTF control codes, and a double backslash shows that you want a backslash character in your text.

The MCI commands create a window that is positioned near the top left corner of the screen, even if that is outside the Viewer window. The window position can be specified by executing another MCI command between the open and play commands. This command would read *put movie destination at X1 Y1 X2 Y2*, where X1 and Y1 are the coordinates of the top left corner of the window, and X2 and Y2 are the width and height of the window. All measurements are in pixels relative to the top-left corner of the display. This command is demonstrated in How-To 9.7.

**Play a Compatible Animation File?**

Complexity: EASY

**Problem**

I want to play an animation file created by Gold Disk's  
 {vfld2305857852620668938}Animation Works Interactive{vfld72057052872  
 048640} in my application.

**Technique**

Gold Disk provides MCI drivers that are compatible with Viewer, so you can use the standard Viewer multimedia controller.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

**Steps**

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP9\_6, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP9\_6. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the movie files for this How To from the VIEWERHT\HOWTOS\CHAP9\CHAP9\_6\MOVIES directory on the CD-ROM to the VIEWERHT\CHAP9\_6\MOVIES subdirectory on your hard drive.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP9\_6 directory. Enter the name of your document file as TEXT\CHAP9\_6.RTF.

Now you can create the document file:

4. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this demonstration doesn't require a context string. Viewer displays the first topic when a file is loaded unless you specify otherwise.
5. Use Topic Editor to insert a Multimedia command, and click on the Options button.
6. Enter {vfld2305865549202063370}GDAnim{vfld-9079242005371944960} as the MCI Device, and select file name MOVIES\CHAP9\_6.AWM. Note that you have to type in the device type—it is not included in the pull-down list.
7. Click on the Layout button, and enter Train Switch as the Caption Text. Click on OK in each dialog box to return to the document.
8. Save the document, and compile and test as usual. The window should look like Figure 9-12.

```
{ewc vwrht2, TsTextButton, "Figure
9½12"[Macro=JI('viewerht.mvb>SecWin', 'fig9_12')] [Font="Arial"
```

/S12/B4] /W100 /H40/B1/D2}

9. Click on the button when it appears, and see the effect.

### How It Works

The multimedia controller you used this time is the same that you used in the previous How-Tos in this chapter. The important difference in this case was forcing an MCI Device name that is not part of the standard Windows 3.1 installation. The device name used must match an entry in the [MCI] section of the SYSTEM.INI file or the command fails. These entries are created as part of installing the appropriate driver software. Related entries are also in the WIN.INI files.

### Comment

Note that the animation file used in this demonstration includes its own pauses with a button to resume play. This uses a feature of the program that created the file, instead of Viewer's controls. Animation software includes many powerful capabilities such as this that help make the image more appealing or informative. This particular file demonstrates the value of animations in training applications.

If you include animation files in your application that require drivers that aren't installed with Windows or sound cards, you must assure that the necessary files and INI file entries are present on your users' systems. This requires including those drivers in your installation package and including instructions that update the INI files in your Setup script. This is described in appendix A. The [MCI] section of my {vfld137438953482}SYSTEM.INI {vfld2850492484943872} file contains the following:

```
{vfld137438953482}Sequencer {vfld2850492484943872} =mciseq.drv
{vfld137438953482}WaveAudio {vfld-
9042665642972413952} =mciwave.drv 7
Mixer =mcimixer.drv
{vfld137438953482}AVIVideo {vfld2850492484943872} =mciavi.drv
{vfld137438953482}CDAudio {vfld2850492484943872} =mciada.drv
{vfld137438953482}GDAnim {vfld2850492484943872} =mciawi.drv
{vfld137438953482}AAAnim {vfld-9151452422936199168} =mciAAP.drv
```

## Play an Incompatible Animation File?

Complexity: DIFFICULT

### Problem

I want to play an animation file in my application, but its MCI driver is not compatible with Viewer.

### Technique

This How-To demonstrates how to play an animation file created by Autodesk {vfld137438953482} Animator Pro {vfld13331578486784}. The {vfld137438953484} Autodesk {vfld-7998112004399169536} MCI driver has a bug that makes it incompatible with the Viewer MVMCI2 command.

You use two techniques to play this file. The first executes external MCI commands directly, since there is an MCI driver for this package. The second executes the Microsoft Media Player (MPlayer) program to play the file.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP9\_7, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP9\_7. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy the movie files for this How To from the VIEWERHT\HOWTOS\CHAP9\CHAP9\_7\MOVIES directory on the CD-ROM to the VIEWERHT\CHAP9\_7\MOVIES subdirectory on your hard drive.
3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP9\_7 directory. Enter the name of your document file as TEXT\CHAP9\_7.RTF.

Next define the external MCI command:

4. Choose Config from the Section menu and paste a RegisterRoutine command into the startup script.
5. Edit the RegisterRoutine command and enter 'mmsystem' in the DLLName field, 'mciSendString' in the FunctionName field, and 'U=SSuu' in the Parameters field. Click on OK in each dialog box, then save the project file.

Now you can create the document file:

6. Use Project Editor to start Word and create your document. You don't need to define a topic—the beginning of the file is always a new topic, and this demonstration doesn't require a context string. Viewer displays the first topic when a file is loaded unless you specify otherwise.
7. Insert a few blank lines, and enter text Play a movie using MCI commands.



8. Select the text you just entered and invoke Topic Editor. Click on OK to accept the Hot Spot (text) entry that is selected.
9. Select the Jump to... element, then click on the Hidden Text is Command(s) radio button.
10. Type in the following commands exactly as shown. Note the single space between the second pair of quotation marks. Enter each command on a separate line.
 

```
{vfld2305865549202063371}mciSendString{vfld-9042521606949175296}("open movies\chap9_7.flc alias movie", " ",0,0)
mciSendString("put movie destination at 15 15 500 500", " ",0,0)
mciSendString("play movie wait", " ",0,0)
mciSendString("close movie", " ",0,0)
```
11. The completed dialog box looks like Figure 9–13. Click on OK to return to the document.

```
{ewc vwrht2, TsTextButton, "Figure
9i½13"[Macro=JI('viewerht.mvb>SecWin', `fig9_13')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

12. Insert a few blank lines, then enter text Play a movie using MPlayer.
13. Repeat steps 6 and 7 to create a hot spot that executes a command using this text.
14. Paste in an ExecProgram command, then click on the Edit Command button. Enter the following in the CommandLine field:
 

```
`{vfld137438953483}MPlayer{vfld2306123943024525312} /play
/close movies\chap9_7.fli'
```
15. Select 0 — Normal from the pull-down list for the ProgramState field. Click on OK in each dialog box to return to the document.
16. Save the document, and compile and test as usual. The window should look like Figure 9–14 after clicking on the first hot spot.

```
{ewc vwrht2, TsTextButton, "Figure
9i½14"[Macro=JI('viewerht.mvb>SecWin', `fig9_14')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

17. The results of clicking on the second hot spot should look like Figure 9–15.

```
{ewc vwrht2, TsTextButton, "Figure
9i½15"[Macro=JI('viewerht.mvb>SecWin', `fig9_15')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

The MCI commands used in the first hot spot are the same as in How-To 9.5,

with the addition of positioning the picture window. The first command opens the movie file and assigns an alias (*movie*) to be used in the following commands. The second command plays the file and waits until it is done before returning to Viewer to execute the next command. The last command closes the file. The second parameter for the `mciSendString` command is always a single space enclosed in quotation marks (" "). The third and fourth parameters are always zero. These parameters provide options that are valuable when the command is called from a C or Visual Basic program, but cannot be used when it is called from a Viewer application. The MCI command is an external command, so it was defined in a `RegisterRoutine` command in steps 4 and 5.

The window position is specified by executing the `put` MCI command between the open and play commands. This command creates the window 25 pixels down and 25 pixels to the right of the upper left corner of the display, and makes the window 150 pixels wide and 150 pixels high.

The wait option with the MCI play command is critical. If it is missing, the next command executes immediately, closing the file before it even begins to play. You probably noticed that you couldn't do anything until the file finished playing.

The `ExecProgram` command you defined in step 12 causes the Microsoft Media Player (MPlayer) program to be run. The `/play` and `/close` options cause the program to begin playing the specified movie automatically, and to exit the program when the movie is finished. A double backslash (`\\`) is needed in the file name because of RTF-file conventions. A single backslash begins certain RTF control codes, and a double backslash shows that the backslash character is desired.

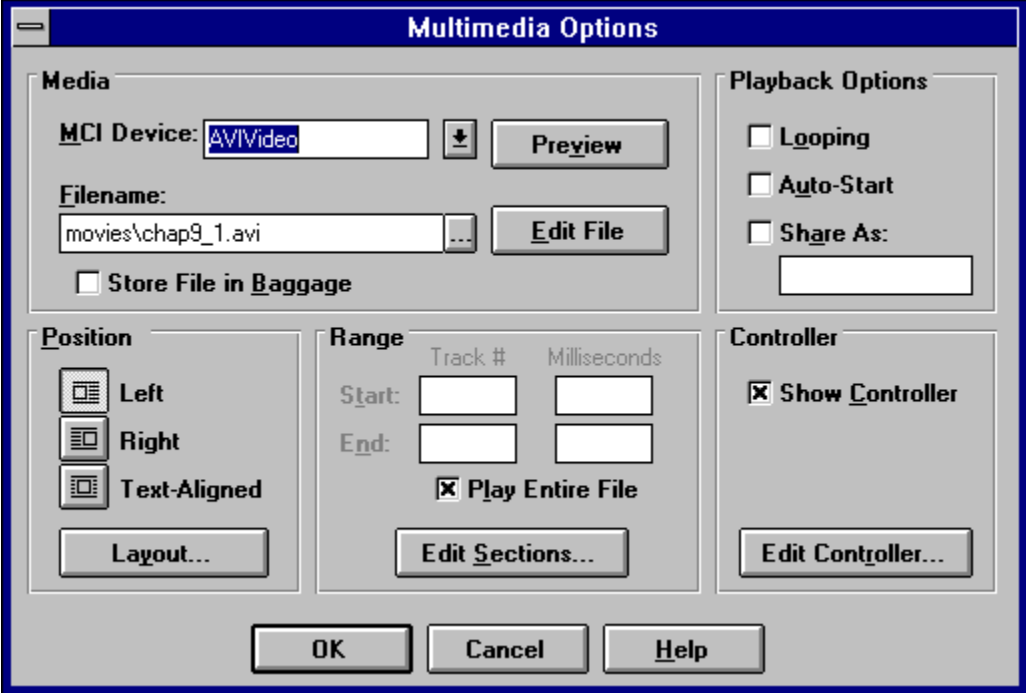
### **Comment**

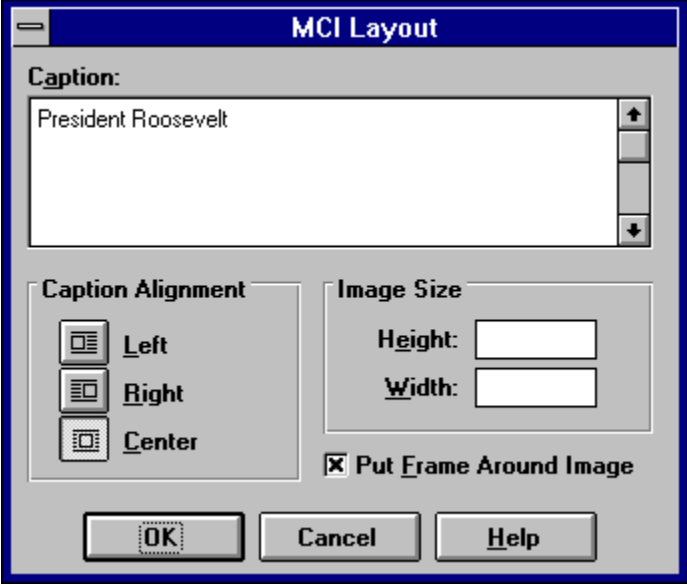
The `ExecProgram` command returns control to Viewer immediately, which lets the topic be displayed while the movie is being played. The delay before the hot spot appears is the time Windows needs to load the MPlayer program. This technique takes longer to begin playing than the multimedia controller because of the additional overhead of executing a separate program.

The only way to use AutoDesk  
{vfld137438953482}FLI{vfld1113255231232} and  
{vfld137438953482}FLC{vfld7363103374608171008} animation files is by converting them to AVI files with the VidEdit program included in Microsoft Video for Windows.

## 9.8 Tips and Tricks

- ⇒ Movies should only be used if you need a video of an actual scene, or closely coordinated sound and pictures. Whenever possible, choose the higher-resolution display of animations.
- ⇒ Continuing improvements in video technology should make high-quality movies with large pictures practical within a few years.
- ⇒ Animation packages have become extremely powerful and easy to use. Some artistic talent is necessary to develop files containing original moving objects (known as actors). If the actors provided with a package are adequate for your needs, very little artistic talent is necessary to produce an attractive result.
- ⇒ Some animation software packages require video systems capable of displaying 256 simultaneous colors, while others can work on either 16- or 256-color systems. Digital movies require 256 colors.
- ⇒ The external commands used in this book, such as `mciSendString`, are documented in packages designed for programmers, such as Microsoft's Software Development Kit and many compilers sold by Microsoft, Borland, and others. This documentation can be very difficult for nonprogrammers to understand. Information and assistance are also available from other sources, as described in appendix C.
- ⇒ Sounds that are included in animation files usually cannot be closely coordinated with the picture. If close synchronization is required, the file must be converted to an AVI movie file before adding the sounds.






**How-To 9.1 Demo**

**File Edit Bookmark Help**


**Contents Go Back History** << >>

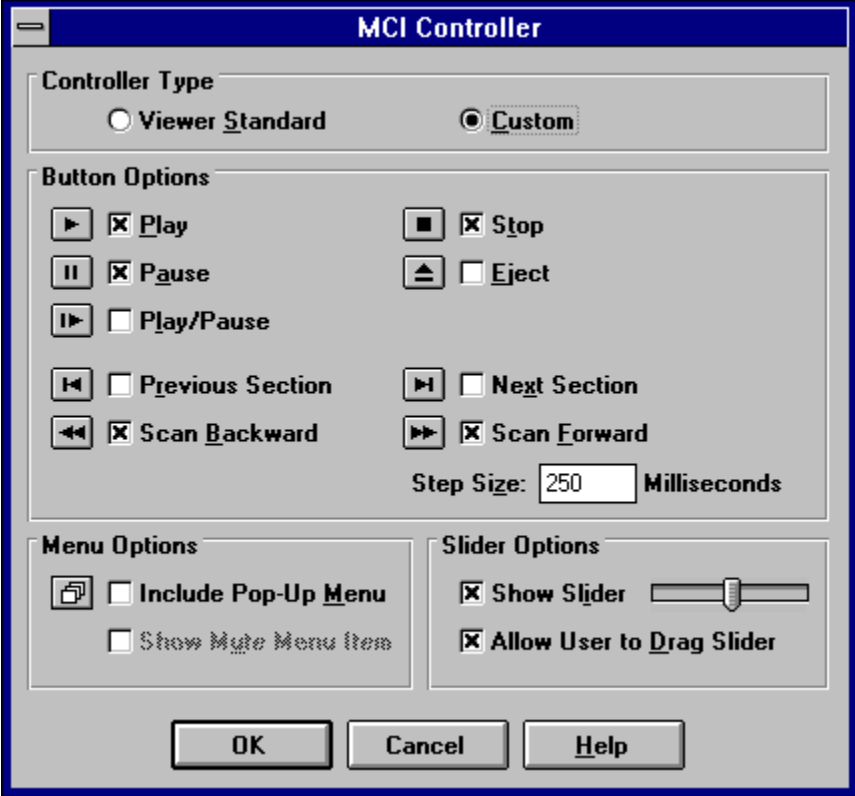
This topic demonstrates the appearance and operation of a standard Viewer multimedia controller playing an AVI movie file.

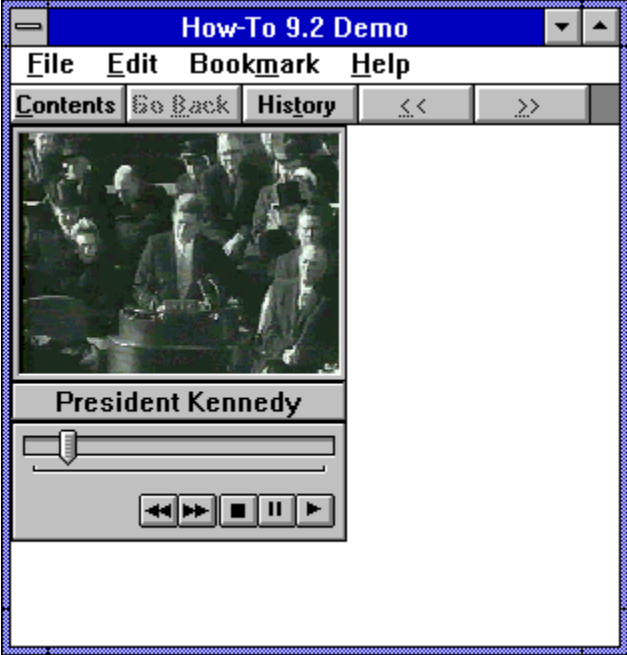


Text is being placed both before and after the controller to show the effect. The controller is left-justified at the beginning of this paragraph.

**President Roosevelt**









**MCI Sections**

**Use Section Information**

**Defined Within the Embedded Pane**

**From a Text File:**  ...

**Store Section Text File in Baggage**

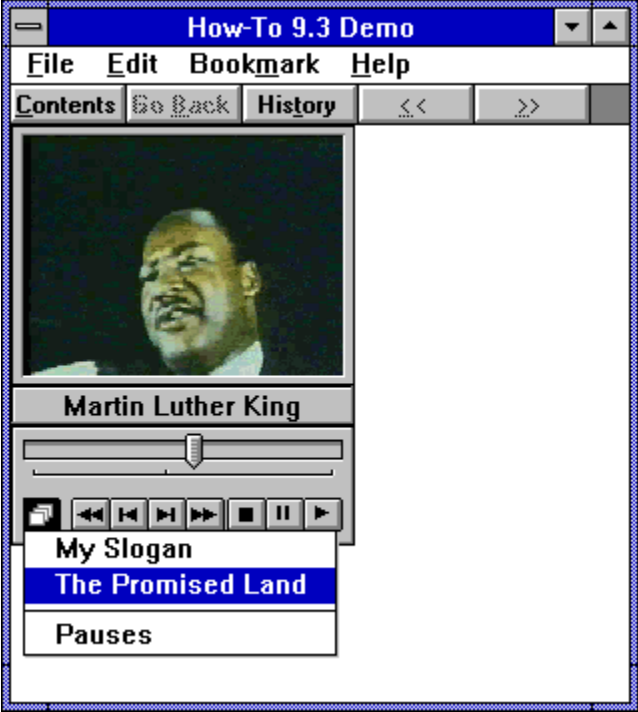
My Slogan  
The Promised Land

**Section Name:**

**Section Begins At:**  
Track #:  Milliseconds:

**Pause at Beginning of Section**

**Display Tick Mark On Slider**




How-To 9.4 Demo

File Edit Bookmark Help

Contents Go Back History << >>

[Go to second topic](#)




The image shows a software window titled "How-To 9.4 Demo". The window has a menu bar with "File", "Edit", "Bookmark", and "Help". Below the menu bar is a toolbar with "Contents", "Go Back", "History", and navigation arrows "<<" and ">>". The main content area contains a green text link "Go to second topic". Below the link is a video player window titled "chap9\_4.avi". The video player shows a scene with a car on a road. The video player has a standard control bar with a play/pause button, a stop button, and a progress slider.

How-To 9.4 Demo

File Edit Bookmark Help

Contents Go Back History << >>



Merging Traffic

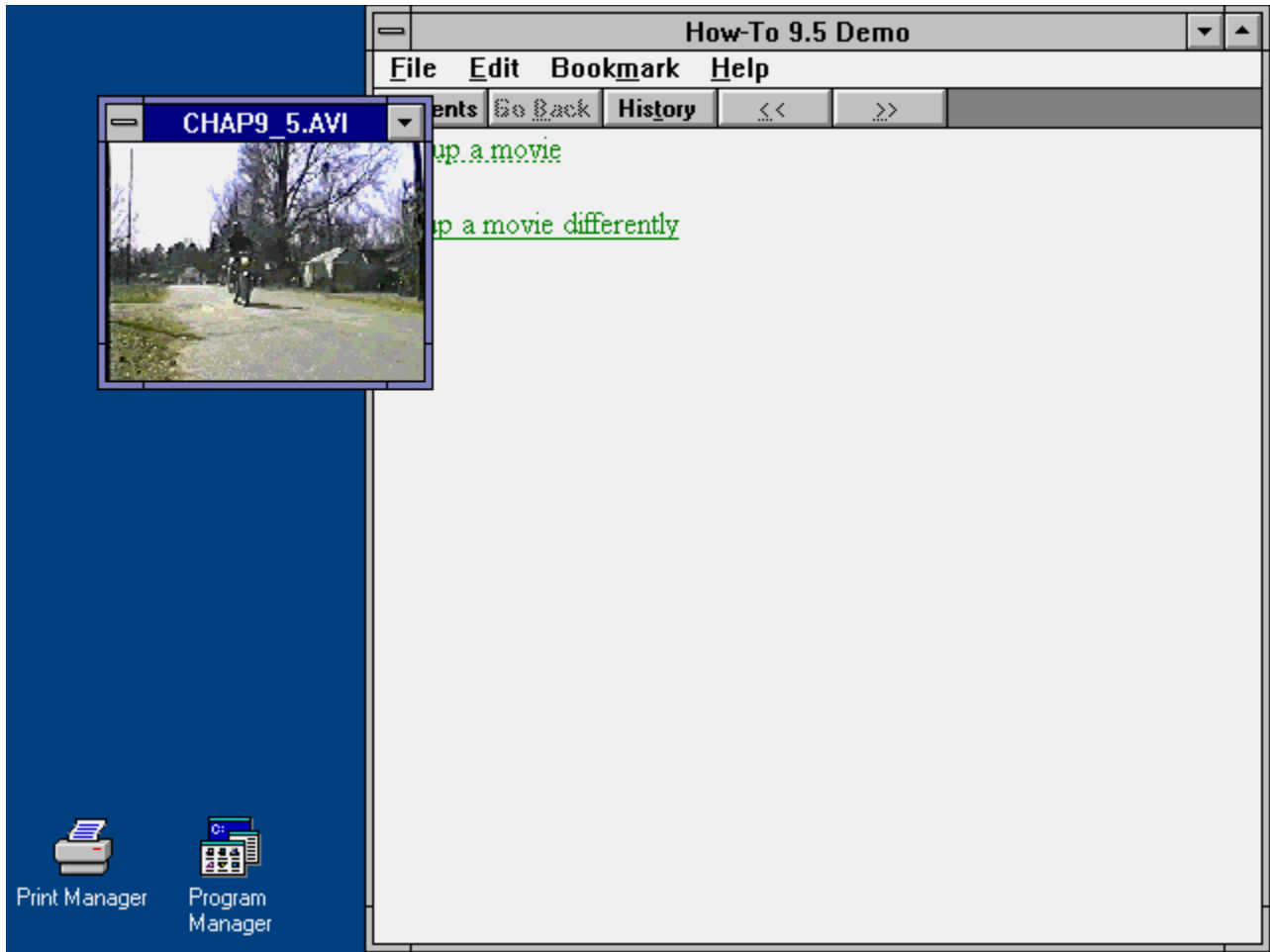
[Go to first topic](#)

[Top up a revie](#)

[Top up](#)



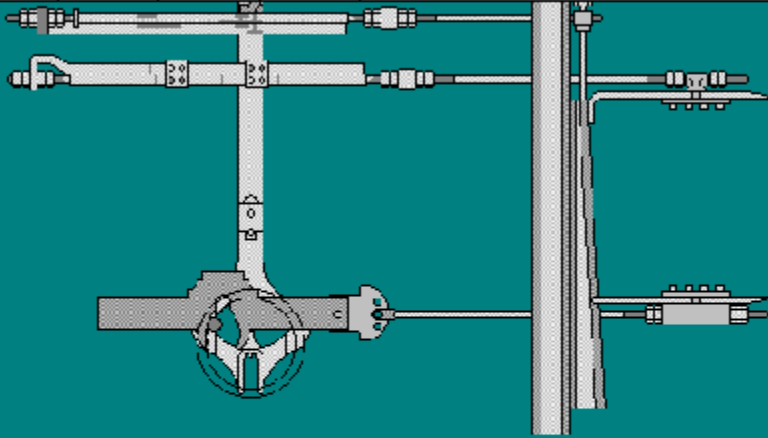
Motorcycle



How-To 9.6 Demo

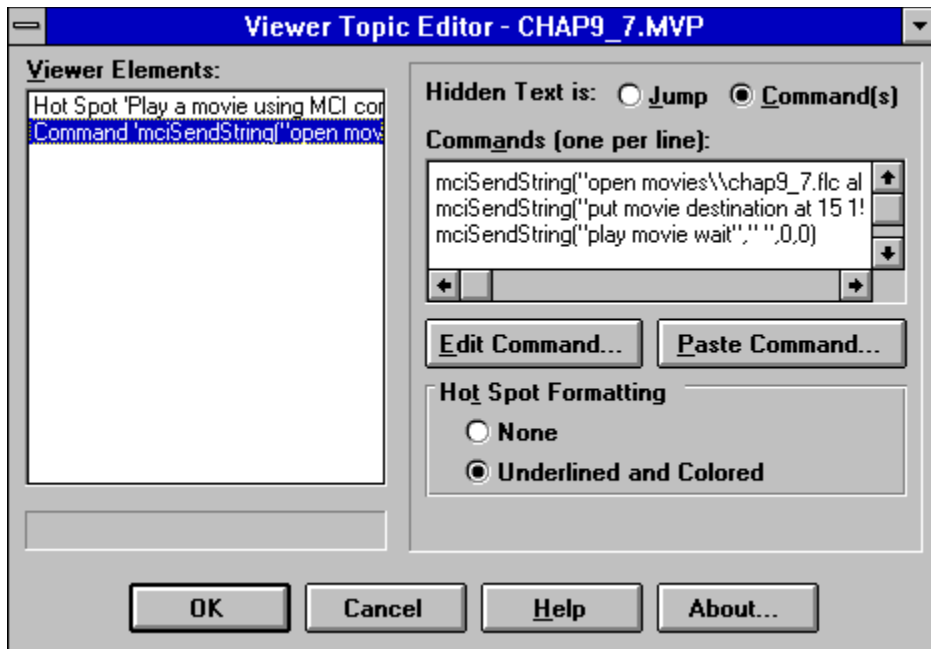
File Edit Bookmark Help

Contents Go Back History << >>

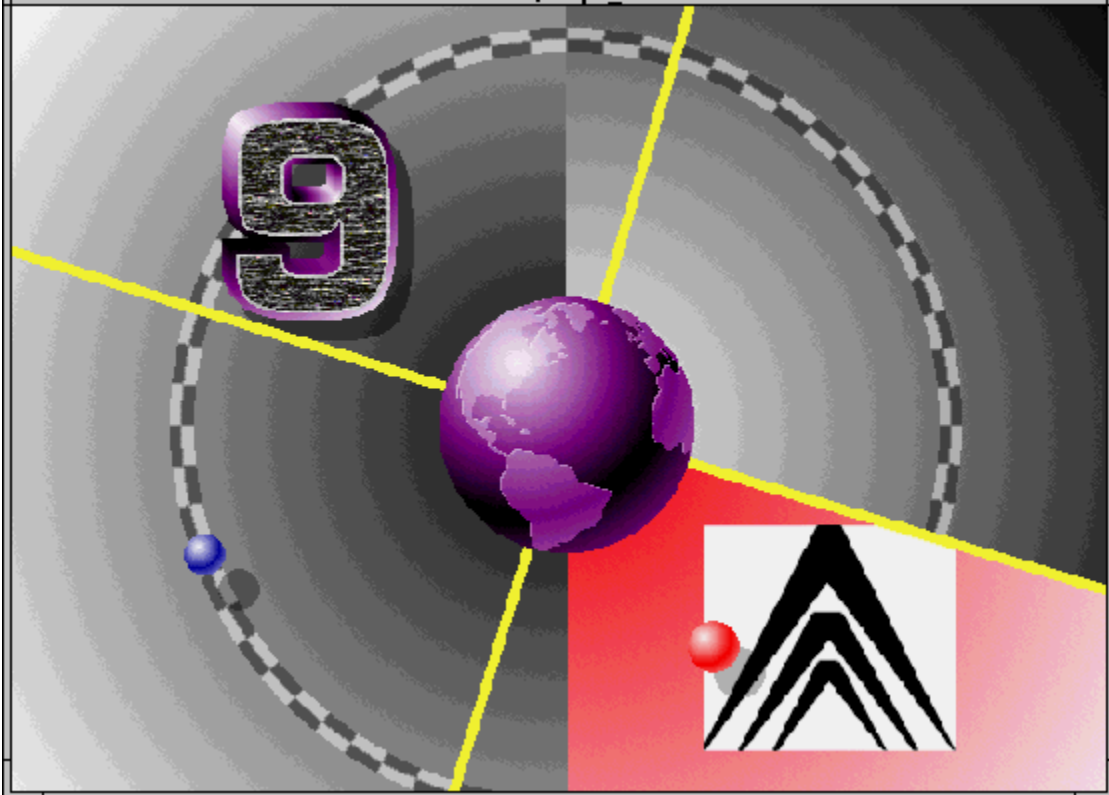


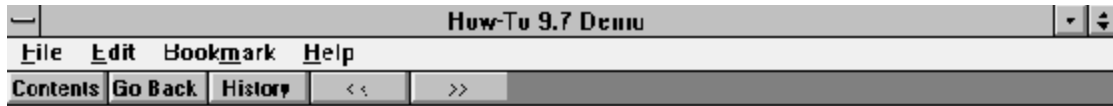
- The cam bar moves and centers the yoke; the cam roller enters the throat of the throw bar.
- The throw bar dog releases the throw bar.
- The point detector yoke goes to the mid position and indication is lost.
- The lock rods are unlocked by the lock rod dog movement.

[Click Here To Continue](#)









Play a movie using MCI commands

Play a movie using





Viewer can do a lot for you without any programming, but sometimes it's useful or even necessary to write programs that work with Viewer. For example:

- ü You may need some functions that aren't available in Viewer.
- ü You may want to use Viewer to provide the Help function for an application, which means your application must tell Viewer which topic to display.
- ü You may want to have a program and a Viewer application working in a coordinated effort to provide the desired services for your users.

There are many opportunities for combining custom programs with Viewer.

Programs can

- ü Issue any Viewer commands, including jumps and popups
- ü Be executed as external commands (in a DLL) by a Viewer application
- ü Create new embedded pane functions such as buttons, list boxes, text input, or reporting the percentage of right answers (score) the user has in a series of multiple-choice questions
- ü Provide a custom Search dialog box and functions that include added capabilities, such as searching across multiple Viewer MVB files
- ü Provide custom word breaks in searching; this could allow phonetic searching, or the use of unique data types
- ü Read files stored in Baggage, for any purpose; you could, for example, let users select a document or spreadsheet through a Viewer application, then extract it as a separate file for input to another program
- ü Perform special operations when the user takes certain actions, such as displaying a new topic, scrolling, or closing the application; the operations could include keeping a second program coordinated with the Viewer application, or assuring that sound files have stopped playing

The possible programs can be combined in many ways. For example, you can write a program that detects when the user minimizes the Viewer window and then issues a command telling Viewer to restore the window to its normal size.

Visual Basic (VB) does not support writing external commands (DLLs), which are needed to take advantage of many of these opportunities. Many of the programs you might want must be written in C or Pascal. VB programs are able to issue Viewer commands, and one of the files supplied on the enclosed CD disk (MV.VBX) makes it possible for a VB program to accept notification of Viewer events such as jumps and popups. How-To 10.2 demonstrates the capabilities of this VB enhancement.

One of the most powerful extension capabilities in Viewer is the ability to execute external commands. Chapters 8 and 9 demonstrate this, by using external commands to play sounds, movies, and animations. For example, a Windows API function is executed to determine if the computer can play sounds. The support for external commands makes it possible for Viewer to

do nearly anything that you can write a program to do!

Custom-written

{vfld137438953482} embedded pane {vfld7232780460391137280} commands let you enhance the user interface while maintaining the appearance of a seamless whole. Viewer comes with a sample program that provides a list box. The CD-ROM enclosed with this book includes an embedded pane command program that let you use your own buttons anywhere in any pane of a main or secondary window. Another program on the disk lets your Viewer application keep track of the user's score on multiple-choice questions, and display the current score in an embedded window. Many commercial Viewer applications use similar functions to provide user capabilities that can't be performed by standard Viewer commands.

The power of these programming capabilities is best understood by considering the Viewer extensions included on the enclosed disk, and the descriptions in Chapter 1 of advanced Viewer applications and other Viewer extensions.

The programs that must be written in C, rather than Visual Basic, are relatively difficult to write. Appendixes C and D explain how to locate experienced programmers to assist you.

## Control Viewer from an Application?

Complexity: INTERMEDIATE

### Problem

I want to use Viewer instead of WinHelp to provide context-sensitive help for my Visual Basic program when the user presses **[F1]**. The Viewer window should be alongside my program's window instead of overlapping.

### Technique

This How-To demonstrates the techniques for controlling a Viewer application from within a VB program. It positions the Viewer window alongside the VB program's window, and instructs Viewer to display selected topics when the user requests context-sensitive help. The user can close the Viewer window, and the program reopens it the next time it's needed.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and create the Viewer files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP10\_1, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP10\_1. Create these four directories for all projects, even if some are not needed.
2. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP10\_1 directory. Enter the name of your document file as TEXT\CHAP10\_1.RTF.
3. Choose Config from the Section menu, and delete all of the commands in the default startup script by selecting them with the mouse and pressing **[DEL]**. Click on OK.
4. Choose Window Definitions from the Sections menu. Enter the value 512 in the Top, Left, Height and Width fields, then click on the Properties button to display the Window Properties dialog box.
5. Enter Demo Help in the Window Caption field. Double-click on the System menu box at the upper-left corner of the window to close this dialog box and return to the Window Definitions dialog box. Click on OK to return to the main Project Editor window.
6. Use Project Editor to start Word and create your document, and create a new topic with context string text1.
7. Enter the following text:

This is the help screen for the first text box.

It could easily contain pictures, sounds, movies, or animations, just like any other Viewer topic.

8. Create a new topic with context string text2, and enter the text:  
This is the help screen for the second text box.

It could easily contain pictures, sounds, movies, or animations, just like any other Viewer topic.

9. Create a new topic with context string button, and enter the text: This is the help screen for the Exit button.

It could easily contain pictures, sounds, movies, or animations, just like any other Viewer topic.

10. Save the document, and compile as usual.

Next create the VB program:

11. Create a new project called CHAP10\_1.MAK, and save it in the \VIEWERHT\CHAP10\_1 subdirectory. Create a new form, filling the left half of the screen, and create the objects and properties listed in Table 10-1. The position of the objects is not critical. Save it as HELPFORM.FRM. The screen looks like Figure 10-1.

**Table 10-1. Application Help Project Objects and Properties**

| Object   | Property  | Setting          |
|----------|-----------|------------------|
| Form     | FormName  | HelpForm         |
|          | Caption   | Viewer Help Demo |
|          | MaxButton | 0 (False)        |
| Text Box | Name      | Text1            |
|          | Height    | 495              |
|          | Left      | 720              |
|          | Text      | First Text Box   |
|          | Top       | 480              |
|          | Width     | 4935             |
| Text Box | Name      | Text2            |
|          | Height    | 495              |
|          | Left      | 720              |
|          | Text      | Second Text Box  |
|          | Top       | 1680             |
|          | Width     | 4935             |
| Button   | Name      | Command1         |
|          | Caption   | Exit             |
|          | Height    | 495              |
|          | Left      | 1560             |
|          | Top       | 3960             |
|          | Width     | 1215             |

```
{ewc vwrht2, TsTextButton, "Figure
10i½1"[Macro=JI('viewerht.mvb>SecWin', `fig10_1')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

12. Create a new module, enter the following code, and save it as HELPFORM.BAS.

' Key Codes

```
Global Const KEY_F1 = &H70
```

```
' ScaleMode  
Global Const TWIPS = 1  
Global Const PIXELS = 3
```

```
' Viewer Constants  
Global Const cmdoptNONE = 0  
Global Const cmdoptHIDE = 1
```

13. Enter the following code in the Declarations section of your form to define the Viewer API functions and common data fields you use. Enter each Declare statement on a single line.

```
Dim ViewerX, ViewerY, ViewerWidth, ViewerHeight As Integer
```

```
Declare Function  
{vfld137438953483}VwrFromMVB{vfld3131967461654528} Lib  
"mvapi2.dll" (ByVal szMVB As String) As Integer  
Declare Function  
{vfld137438953483}VwrCommand{vfld3131967461654528} Lib  
"mvapi2.dll" (ByVal iVwr As Integer, ByVal szMVB As String, ByVal  
szMacro As String, ByVal iCmdOptions As Integer) As Integer  
Declare Function  
{vfld137438953483}VwrQuit{vfld280933810831360} Lib  
"mvapi2.dll" (ByVal iVwr As Integer) As Integer
```

14. Enter the following code in the Command1\_Click event subroutine. This terminates Viewer, if it is still running, and ends the VB program.

```
Sub Command1_Click ()  
  
' end Viewer  
MVB$ = "CHAP10_1.MVB" + Chr$(0)  
vwrViewer% = VwrFromMVB(MVB$)  
If vwrViewer% > 0 Then  
Ret% = VwrQuit(vwrViewer%)  
End If  
  
' end VB program  
Unload HelpForm  
End
```

```
End Sub
```

15. Enter the following code in the Command1\_KeyDown, Text1\_KeyDown, and Text2\_KeyDown event subroutines to handle a user's request for help (by pressing **[F1]**).

```
Sub Command1_KeyDown (KeyCode As Integer, Shift As Integer)  
If KeyCode = KEY_F1 Then  
ContextHelp ("button")  
End If  
End Sub
```

```
Sub Text1_KeyDown (KeyCode As Integer, Shift As Integer)  
If KeyCode = KEY_F1 Then  
ContextHelp ("text1")  
End If  
End Sub
```

```
Sub Text2_KeyDown (KeyCode As Integer, Shift As Integer)  
If KeyCode = KEY_F1 Then  
ContextHelp ("text2")  
End If
```



End Sub

16. Enter the following code in the Form\_Load event subroutine. This positions the VB program's window to occupy the left half of the screen, and calculates the position that the Viewer window should occupy.

```
Sub Form_Load ()  
  
    ' set this window's size & position  
    Width = Screen.Width / 2  
    Height = Screen.Height  
    Top = 0  
    Left = 0  
  
    ' calculate Viewer windows's size & position, in pixels  
    ScaleMode = TWIPS  
    TwipHeight = ScaleHeight  
    TwipWidth = ScaleWidth  
    ScaleMode = PIXELS  
    PixelHeight = ScaleHeight  
    PixelWidth = ScaleWidth  
  
    ViewerX = ((Width * PixelWidth) / TwipWidth)  
    ViewerY = 0  
    ViewerWidth = ViewerX - 1  
    ViewerHeight = ((Height * PixelHeight) / TwipHeight)  
  
End Sub
```

17. Enter the following code as a new subroutine. This issues the Viewer API calls to load Viewer and display the proper topic.

```
Sub ContextHelp (Context As String)  
    ' prepare strings  
    MVB$ = "CHAP10_1.MVB"  
    Q$ = Chr$(34)  
  
    ' create Viewer PositionTopic command  
    PosCmd$ = "PositionTopic(" + Q$ + "main" + Q$ + ","  
    PosCmd$ = PosCmd$ + Format$(ViewerX) + ","  
    PosCmd$ = PosCmd$ + Format$(ViewerY) + ","  
    PosCmd$ = PosCmd$ + Format$(ViewerWidth) + ","  
    PosCmd$ = PosCmd$ + Format$(ViewerHeight) + ",1,0)"  
  
    ' create Viewer Jump command  
    JICmd$ = "JI(" + Q$ + MVB$ + Q$ + "," + Q$ + Context + Q$ +  
    ")"  
  
    ' get identifier for Viewer task  
    vwrViewer% = VwrFromMVB(MVB$ + Chr$(0))  
  
    ' Issue command  
    ' note: if Viewer is not running, vwrViewer% = 0  
    ' This causes VwrCommand to start Viewer  
  
    vwrRet% = VwrCommand(vwrViewer%, MVB$ + Chr$(0),  
    PosCmd$ + ";" + JICmd$ + Chr$(0), cmdoptNONE)  
  
End Sub
```

18. Save the project files, and compile them to create an EXE file named CHAP10\_1.EXE in the \VIEWERHT\CHAP10\_1 directory on your hard drive, then exit VB.

19. Execute the compiled program. Select an object on the form and press **[F1]**. The screen should look like Figure 10–2. Repeat this for each object. Try closing the Viewer window in between by double-clicking on the System menu box at the upper left corner of the window. Click on the Exit button to end the test.

```
{ewc vwrht2, TsTextButton, "Figure  
10i½2"[Macro=JI('viewerht.mvb>SecWin', `fig10_2')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

The `Form_Load` subroutine assures that the windows won't overlap by forcing the VB program's window to occupy the left half of the screen, and by calculating the desired size and position of the Viewer window in the right half. This gets a little tricky because Viewer's window-positioning command requires that the values be in pixels, but VB doesn't give access to the size of its complete window in pixels—only the size of the user area. This routine gets the size of the user area in both twips and pixels, then applies the pixel/twip ratio for the current screen to the window's size and position.

The `KeyDown` subroutines call the common `ContextHelp` subroutine, passing the context string of the desired Viewer topic.

The `ContextHelp` subroutine first creates strings containing the `Viewer PositionTopic` and `JumpID` (in the abbreviated `JI` form) commands, using the same format those commands use in a Viewer topic. Then you execute the `VwrFromMVB` API function to get a special Viewer-instance identifier. This function returns zero if a Viewer task isn't running for the specified file. The `VwrCommand` function is then called with the instance identifier. A value of zero instructs this function to start a Viewer session for the specified file.

You probably noticed that the Viewer window appears for a moment in the lower-right corner of the screen before it expands to fill the desired area. This initial appearance comes from the values you entered in the `Top`, `Left`, `Height`, and `Width` fields in Project Editor's `Window Properties` dialog box. If you can predict the size and position you want, enter the proper values in those fields to avoid the two-stage appearance of the Viewer window.

The `Command1_Click` subroutine works similarly to the `ContextHelp` routine. If `VwrFromMVB` returns a nonzero value, the `VwrQuit` function is called to end the Viewer session. There's no need to do this if a zero is returned, of course—that means the user already ended the session.

The Viewer document file is straightforward. You create one topic for each subject, with a context string that matches those in the VB program. In this example you don't provide the user with any means of navigating between topics—each topic stands alone. You delete the default `Config` script in step 3 to eliminate the standard buttons and menus as part of this effort to make each topic unique. In a real help file you might allow normal navigation, and provide a table of contents, browse groups, and other navigation aids. The Viewer application would probably also include sounds or other multimedia functions that `WinHelp` doesn't support easily.

### Comment

In a more complex program, the `HelpContextID` values could be passed to

the common ContextHelp routine. They would determine the context string as the index into a table or by converting directly to a string in the form topic\_nn.

Visual Basic comes with sample programs such as CallHelp and IconWorks that demonstrate techniques for calling WinHelp. Although the VwrCommand function does not support the options available in WinHelp, you can easily implement these options by executing the appropriate Viewer command in place of JumpID when executing the VwrCommand function.

## Coordinate a Program with Viewer?

Complexity: ADVANCED

### Problem

I want to use VB and Viewer together to produce an integrated application, using the abilities of each one as appropriate. This means I need to have both windows reflect user actions in either window.

### Technique

You use the Visual Basic custom control (MV. {vfld137438953482}VBX{vfld-9223091103043944448}) included in the CD-ROM enclosed with this book to make the VB program aware of user actions in the Viewer window. The VB program issues commands to the Viewer window based on actions taken in the VB window.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and create the Viewer files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP10\_2, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP10\_2. Create these four directories for all projects, even if some are not needed.
2. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP10\_2 directory. Enter the name of your document file as TEXT\CHAP10\_2.RTF.
3. Choose Window Definitions from the Section menu, then click on the Properties button. Enter 512 in the Left and Width fields, and Viewer Window in the Window Caption field. Double-click on the System menu box to close this dialog box, then click on OK to return to the main Project Editor window.
4. Choose Title Options from the Section menu, and enter contents in the Contents Topic field. Click on OK.
5. Choose Config from the Section menu, and paste in a RegisterRoutine command. Edit that command and enter `VBMVLINK' in the DLLName field, `BroadcastMessages' in the FunctionName field, and `U' in the ParameterSpec field.
6. Type BroadcastMessages(hwndApp) as the next command in the Config script. External commands can't be in the pull-down list because they are unknown to Viewer. Click on OK.
7. Choose Groups from the Section menu, then click on New to define a new group named Group1. Clear the Searchable check box, then click on OK.
8. Save your updated project file.
9. Use Project Editor to start Word and create your document, and create a new topic with context string contents. Be sure to delete the page break created before your topic.

10. Use Topic Editor to create a Topic group footnote, specifying Topic Browse Sequence Group1 and Sequence Number 005.
11. Use Topic Editor to create a Title footnote, specifying Contents as the title.
12. Enter text:  
This is the table of contents topic. Click on a hot spot or use the Browse buttons to display a different topic.
13. Insert a couple of blank lines, then define three hot spots with text Jump to topic 1, Jump to topic 2, and Jump to topic 3. These hot spots should refer to context strings topic\_1, topic\_2, and topic\_3 respectively.
14. Define a new topic with context string topic\_1, Browse Sequence Group1 and Sequence Number 010, and title Topic 1. Enter text:  
This is topic number 1.
15. Create topic 2 and topic 3 similar to topic 1.
16. Save the document and compile as usual.

Next create the VB program:

17. Create a new project called CHAP10\_2.MAK, and save it in the \VIEWERHT\CHAP10\_2 subdirectory. Create a new form, filling the left half of the screen, and create the objects and properties listed in Table 10-2. The position of the objects is not critical.

**Table 10-2. Coordinated Project Objects and Properties**

| Object | Property  | Setting          |
|--------|-----------|------------------|
| Form   | FormName  | Chap10_2         |
|        | Caption   | VB Window        |
|        | MaxButton | 0 (False)        |
| Button | Name      | Command1         |
|        | Index     | 0                |
|        | Caption   | Display Contents |
|        | Left      | 360              |
|        | Height    | 495              |
|        | Top       | 1920             |
|        | Width     | 2055             |
| Button | Name      | Command1         |
|        | Index     | 1                |
|        | Caption   | Display topic 1  |
|        | Left      | 360              |
|        | Height    | 495              |
|        | Top       | 2640             |
|        | Width     | 2055             |
| Button | Name      | Command1         |

|           |         |                     |
|-----------|---------|---------------------|
|           | Index   | 2                   |
|           | Caption | Display topic 2     |
|           | Left    | 360                 |
|           | Height  | 495                 |
|           | Top     | 3360                |
|           | Width   | 2055                |
| Button    | Name    | Command1            |
|           | Index   | 3                   |
|           | Caption | Display topic 3     |
|           | Left    | 360                 |
|           | Height  | 495                 |
|           | Top     | 4080                |
|           | Width   | 2055                |
| Button    | Name    | Command2            |
|           | Caption | Exit                |
|           | Left    | 3480                |
|           | Height  | 495                 |
|           | Top     | 3000                |
|           | Width   | 1215                |
| Label box | Name    | Label1              |
|           | Caption | Last Viewer action: |
|           | Height  | 375                 |
|           | Left    | 360                 |
|           | Top     | 240                 |
|           | Width   | 1800                |
| Label box | Name    | Label2              |
|           | Caption | (none}              |
|           | Height  | 495                 |
|           | Left    | 360                 |
|           | Top     | 840                 |
|           | Width   | 4455                |

---

18. Choose Add File from the File menu, and select the MV.VBX file in your Windows System subdirectory, then double-click on the new icon in the Toolbox to add the control to your program. The screen should look like Figure 10-3. Save the form as CHAP10\_2.FRM.

```
{ewc vwrht2, TsTextButton, "Figure
10i½3"[Macro=JI('viewerht.mvb>SecWin', `fig10_3')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

19. Create a new module, enter the following code, and save it as

```
CHAP10_2.BAS:
' ScaleMode
Global Const TWIPS = 1
Global Const PIXELS = 3
```

```
' Viewer Constants
Global Const cmdoptNONE = 0
Global Const cmdoptHIDE = 1
```

20. Enter the following code in the Declarations section of your form:

```
Dim VPos As String
Dim vwrViewer As Integer
Dim ViewerX, ViewerY, ViewerWidth, ViewerHeight As Integer
Dim JumpRequested, EndRequested As Integer
```

```
Declare Function VwrQuit Lib "mvapi2.dll" (ByVal iVwr As Integer) As Integer
```

21. Enter the following code in the Command2\_Click event subroutine to terminate Viewer and the VB program:

```
Sub Command2_Click ()
' end Viewer
EndRequested = True
Ret% = VwrQuit(vwrViewer)
' end VB Program
Unload chap10_2
End
End Sub
```

22. Enter the following code in the Command1\_Click event subroutine to cause Viewer to display the requested topic:

```
Sub Command1_Click (Index As Integer)
JumpRequested = True
Q$ = Chr$(34)
Label2.Caption = "Jump requested"

MV1.ViewerTitle="CHAP10_2.MVB"
MV1.Vwr = vwrViewer
Cmd$ = "Jl(" + Q$ + "CHAP10_2.MVB" + Q$ + "," + Q$

If Index = 0 Then
Cmd$ = Cmd$ + "contents"
Else
Cmd$ = Cmd$ + "topic_" + Format$(Index)
End If

Cmd$ = Cmd$ + Q$ + ")"
MV1.ViewerCommand = Cmd$
MV1.ViewerExecute = True

End Sub
```

23. Enter the following code in the Form\_Load event subroutine to position the VB program's window in the left half of the screen, calculate the position that the Viewer window should occupy, and start Viewer:

```
Sub Form_Load ()
' set this window's size & position
Width = Screen.Width / 2
```

```

Height = Screen.Height
Top = 0
Left = 0

' calculate Viewer windows's size & position, in pixels
ScaleMode = TWIPS
TwipHeight = ScaleHeight
TwipWidth = ScaleWidth
ScaleMode = PIXELS
PixelHeight = ScaleHeight
PixelWidth = ScaleWidth

ViewerX = ((Width * PixelWidth) / TwipWidth)
ViewerY = 0
ViewerWidth = ViewerX - 1
ViewerHeight = ((Height * PixelHeight) / TwipHeight)

Q$ = Chr$(34)
VPos = "PositionTopic(" + Q$ + "main" + Q$ + ","
VPos = VPos + Format$(ViewerX) + ","
VPos = VPos + Format$(ViewerY) + ","
VPos = VPos + Format$(ViewerWidth) + ","
VPos = VPos + Format$(ViewerHeight) + ",1,0)"

JumpRequested = False
EndRequested = False
Label2.Caption = "File Load"

Show

MV1.ViewerTitle = "CHAP10_2.MVB"
MV1.Vwr = NULL
MV1.ViewerCommand = VPos
MV1.ViewerExecute = True
vwrViewer = MV1.Vwr

```

End Sub

24. Enter the following code in the MV1\_EndJump event subroutine to respond to Viewer topic jumps:

```

Sub MV1_EndJump (vwr As Integer, TopicAddress As Long,
ScrollPosition As Long)

If vwr = vwrViewer Then
    If JumpRequested Then
        JumpRequested = False
        Label2.Caption = "Jump to topic " + Format$(TopicAddress,
"#,##0")
    Else
        MV1.ViewerTitle = "CHAP10_2.MVB"
        MV1.Vwr = vwrViewer
        MV1.ViewerCommand = "Back()" ' Undo the jump
        MV1.ViewerExecute = True
    End If
End If

End Sub

```

25. Enter the following code in the MV1\_MinMax event subroutine to respond to minimizing or maximizing the Viewer window:

```

Sub MV1_MinMax (vwr As Integer, MinMaxState As Long)

```



```

If vwr = vwrViewer Then
    MV1.ViewerTitle = "CHAP10_2.MVB"
    MV1.Vwr = vwrViewer
    MV1.ViewerCommand = VPos
    MV1.ViewerExecute = True
    Label2.Caption = "Size restored"
End If

```

End Sub

26. Enter the following code in the MV1\_Size event subroutine to respond to resizing the Viewer window:

```

Sub MV1_Size (vwr As Integer, iWidth As Long, iHeight As Long)

```

```

If vwr = vwrViewer Then
If (iWidth <> ViewerWidth) or (iHeight <> ViewerHeight) Then
    MV1.ViewerTitle = "CHAP10_2.MVB"
    MV1.Vwr = vwrViewer
    MV1.ViewerCommand = VPos
    MV1.ViewerExecute = True
    Label2.Caption = "Size restored"
End If

```

End If

End Sub

27. Enter the following code in the MV1\_Term event subroutine to respond to terminating the Viewer session:

```

Sub MV1_Term (vwr As Integer, hViewerInst As Long)

```

```

If vwr = vwrViewer Then
    If Not EndRequested Then
        StartRequested = True
        MV1.ViewerTitle = "CHAP10_2.MVB"
        MV1.Vwr = NULL
        MV1.ViewerCommand = VPos
        MV1.ViewerExecute = True
        Label2.Caption = "Viewer restarted"
    End If

```

End If

End Sub

28. Save the project files, and compile them to create an EXE file named CHAP10\_2.EXE in the \VIEWERHT\CHAP10\_2 directory on your hard drive, then exit VB.
29. Execute the compiled program, and the Viewer window should appear in a few moments. Try using the Viewer hot spots or Browse buttons to jump between topics—these jumps should be immediately undone by the VB program. Click on the VB buttons, and the requested topic should appear in the Viewer window. The screen should look like Figure 10–4. Use the Min and Max buttons on the Viewer window. Does the VB program restore the normal window properly in each case? Click on the Exit button in the VB window to end both programs.

```

{ewc vwrht2, TsTextButton, "Figure
10i½4"[Macro=JI('viewerht.mvb>SecWin', `fig10_4')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}

```

## How It Works

The Viewer files are straightforward, with the addition of defining and executing the BroadcastMessages external command in steps 5 and 6. This command activates the link to the VB custom control (MV.VBX), and causes Viewer events to be reported to the VB program. These events are handled by a number of new VB event subroutines.

The Command2\_Click subroutine is almost exactly the same as in How-To10.1. It assumes that a Viewer session is running because other routines restart the session if it is ended outside this routine. The EndRequested flag is set to prevent such a restart.

The Command1\_Click subroutine uses the custom control properties to issue a Viewer jump command to the requested topic. The JumpRequested flag is set to show that this jump should not be reversed when it is reported back to this program.

The Form\_Load subroutine is almost identical to the routine in How-To10.1. It uses the custom control properties to issue the command to start Viewer, initializes the JumpRequested and EndRequested flags, and saves the Viewer instance identifier that other subroutines use to identify messages from the proper application.

The EndJump subroutine executes every time the Viewer application displays a different topic. It first checks if the Viewer instance reporting this jump is the one this program is controlling. If not, this subroutine does nothing. If the jump is not requested from within the VB program, a Viewer Back() command is executed to reverse the jump. Otherwise the JumpRequested flag is turned off and the text box is updated. The Topic Address value returned by Viewer is included in the text box message.

The MinMax subroutine executes when the Viewer window is minimized or maximized. It checks the Viewer instance, and ignores messages from other applications. The Viewer PositionTopic command returns the window to its desired size and position. This command is successful if the window is maximized, but not if it is minimized. This subroutine should use Windows APIs to restore the Viewer window. The Size subroutine executes the same PositionTopic command unless the window is being restored to its normal size and position.

The Term subroutine executes when the Viewer session is ending. If this is not requested within the VB program, the Viewer session is restarted. This subroutine never actually does anything because the Viewer instance being terminated never matches the saved value—it is always reported as zero.

If two or more Viewer sessions are executing at the same time, each of them reports its events to any program that is using this VBX. All of your VBX event subroutines *must* compare the Viewer instance value for the current event to the value for the Viewer session you are trying to monitor or control. This lets your program handle events from the right session while ignoring any others. Some non-Viewer applications may also cause some of these events to be reported with an instance value of zero.

## Comment

You can make a simple and interesting enhancement to this demonstration by

defining another button in the VB program with the caption Allow A Jump. The Click event for that button should turn on the JumpRequested flag, but not issue a Viewer command. This lets you make one jump in the Viewer window that is not reversed, but is reported in the VB text box. This should help you understand the control you can have over a Viewer window from within a VB program.

The topic numbers reported in the BeginJump and EndJump subroutines do not have any obvious meaning—they are not sequential numbers or any other obvious intent. The values may change when the Viewer application is recompiled. These characteristics make them difficult to use. You should store the values in a table in your program so that they can be updated easily.

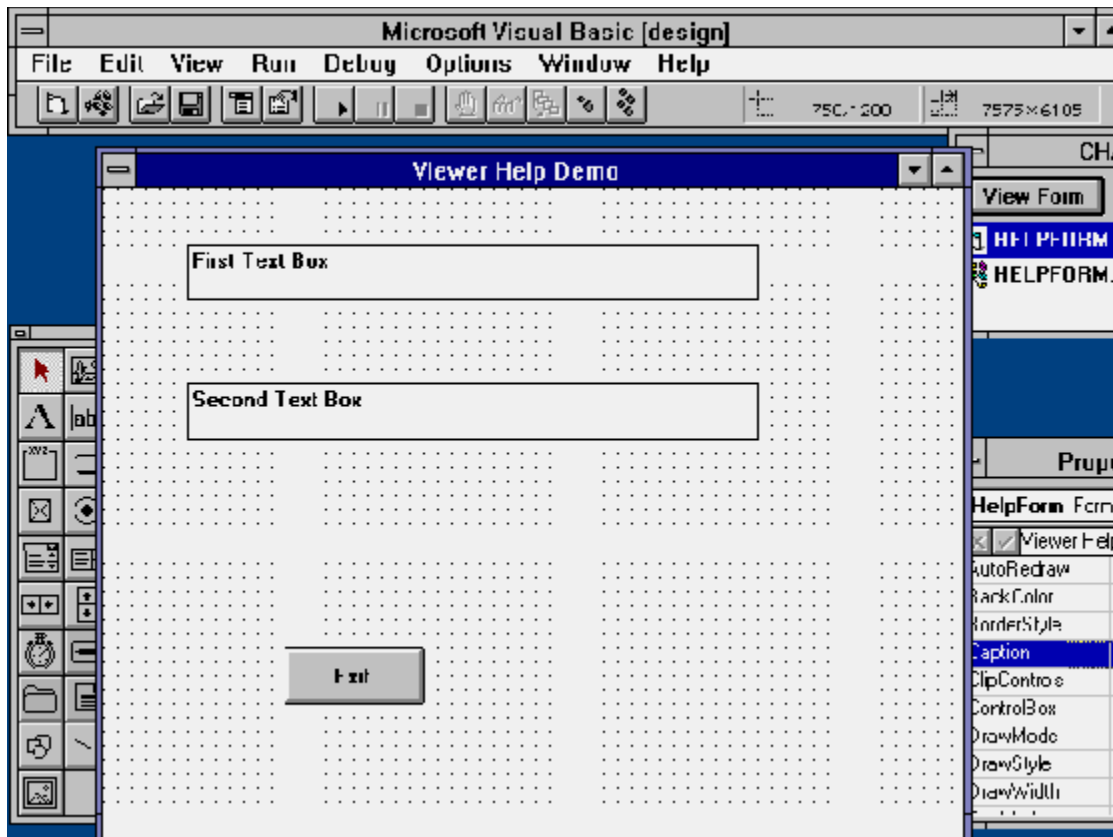
### 10.3 Tips and Tricks

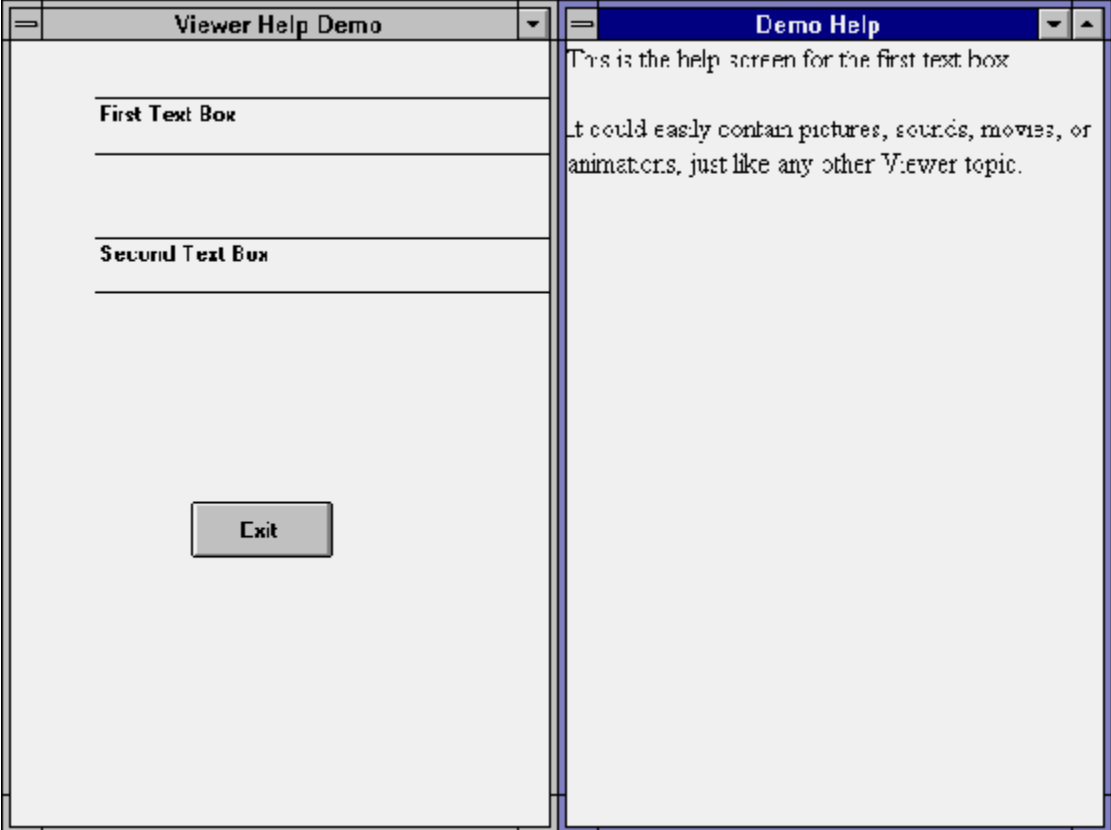
- ⇒ Viewer commands can also be issued by Word for Windows by using {vfld137438953482} WordBasic{vfld72057052872048640}. For example, the following section of code issues a Jump command to Viewer:

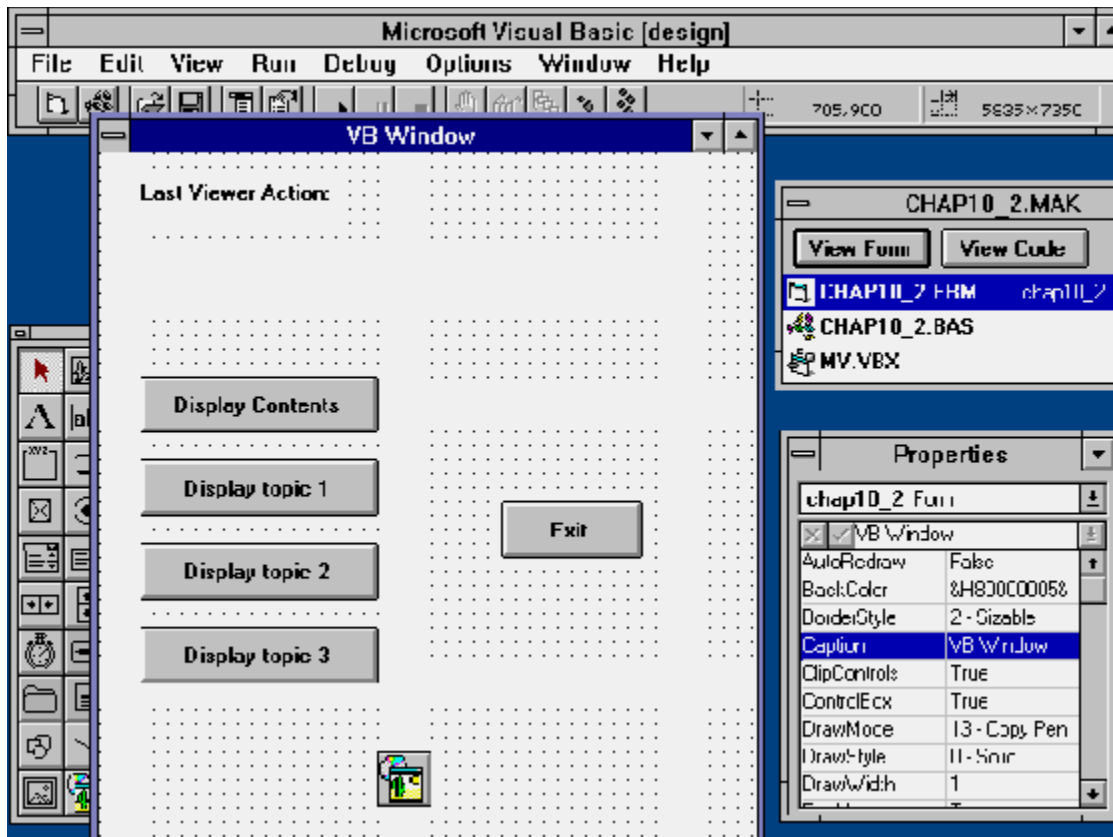
```
Declare Function VwrCommand Lib "mvapi2.dll" (vwr As Integer,
MVB$, Command$, optCmd As Integer) As Integer
```

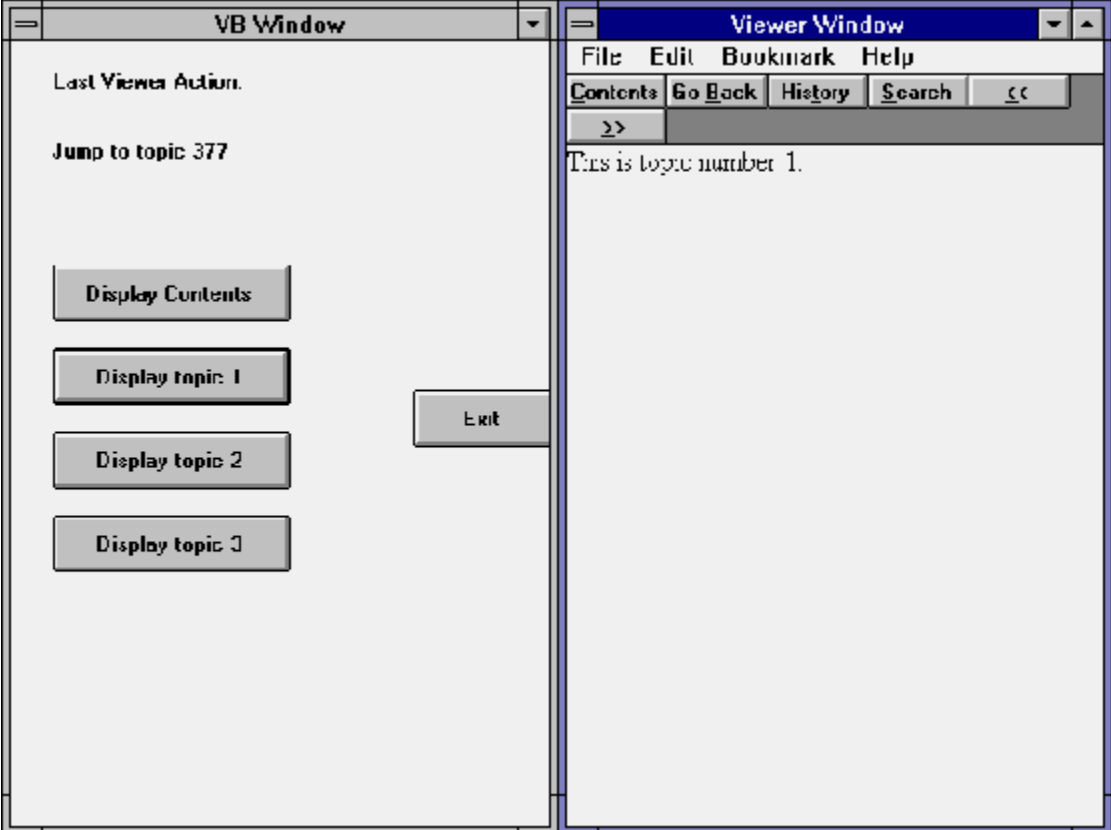
```
Sub Main
    i = VwrCommand(NULL, "CHAP10_2.MVB",
        "JI('CHAP10_2.MVB', `topic_2')", 0)
End Sub
```

- ⇒ The Viewer variable {vfld137438953482} hwndContext{vfld-9223356093936173056} is defined incorrectly in the Viewer documentation. It is the Windows handle of the current window's master pane, not of the window itself.
- ⇒ The lParam1 and lParam2 parameters for the DW\_SIZE message are described incorrectly in the Viewer documentation. The LOWORD of lParam1 contains the width of the main window, and the HIWORD contains the height. lParam2 is always zero.
- ⇒ Viewer 1.0 used {vfld137438953484} hdc{vfld-9042384167995703296} as the name of a field in the RenderInfo structure. This has been renamed as hds in Viewer 2.0. You must change this if you convert an old program.
- ⇒ The {vfld137438953484} PositionWindow{vfld-9223356093936173056} command in Viewer 1.0 and WinHelp uses all of the SDK's SW\_ values for the window state, although the documentation states that only 0, 1, and 2 are allowed. Viewer 2.0 actually *does* use only the documented values of 0, 1, and 2. You must change older programs accordingly to use them with Viewer 2.0.
- ⇒ All Viewer {vfld137438953482} handle{vfld-9042102693018992640} variables are 32-bit values, in preparation for Windows NT and other future releases. Although Windows 3.1 API functions expect 16-bit handles, they work properly if the handle is the first parameter. The method used to pass these parameters assures that the extra 16 bits are not seen by the API. For example, you can use the MessageBox function to display author-defined error messages: define the function by including  
RegisterRoutine("user", "MessageBox", "USSu")  
in your Config script, and executing the external command with  
{vfld2305865549202063371} MessageBox{vfld-9079242005371944960}(hwndApp, "message", "title", 0).













When you design a large application, you must make it easy for the user to locate desired information. You do this, in part, by placing information into a logical hierarchy of topics and groups, and giving the user a clear path through carefully placed hot spots. Sometimes this isn't enough—the user may not know enough to identify the topic or group containing the needed information, or the information may be spread over a number of different topics. Then the user must depend on Viewer's Index and Search functions.

The Index function corresponds to the index in a book. The user looks up a word in the list provided and gets a list of associated topics. The user then selects one of those topics, and Viewer displays it. The index is created when you define keywords in your topics, as demonstrated in Chapter 3. You can define keywords anywhere within a topic, not only at the beginning. The topic displays from the beginning of the paragraph containing the selected keyword. You can also use more than one keyword list. This can be useful if you want to divide a long list of keywords into logical categories. How-To 11.1 demonstrates this technique.

One of the greatest strengths of Viewer is its powerful searching capability and the degree of control it gives you. In its simplest form, this lets the user search for any word in the entire text of your application. The user can search for words or phrases anywhere in the text, or limit the search to topic groups or text categories you define. The search criteria can combine terms with logical operators (such as *and*, *or*, and *not*) to refine the search. Viewer creates the Search dialog box dynamically, based on the capabilities you define. Figure 11-1 shows a Search dialog box showing all of the standard options.

```
{ewc vwrht2, TsTextButton, "Figure  
11-1"} [Macro=JI('viewerht.mvb>SecWin', 'fig11_1')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Viewer lets the author control and customize the Search function in many ways—all with the goal of making it easier for the user to find the desired information. You can

- ü Define search  
{vfld137438953482}fields{vfld4363420484763648}, which are categories of information containing a group of words or phrases just like index keywords. The user can select from a list of entries. How-To 11.2 demonstrates the use of fields.
- ü Use numeric data, as well as characters, within fields. This lets the user search for values such as dates or numbers. You define fields such as birthdates, ages, or weights that can be used in the search criteria. The numbers must be in the proper format to be interpreted properly. How-To 11.2 describes the standard formats supported by Viewer.
- ü Use {vfld137438953482}aliases{vfld-9223356093936173056} to define alternate versions of words or phrases in your text. This could include alternate names (*Samuel Clemens* for *Mark Twain*) or alternate formats of text required for numeric fields (*07/04/1776* for *July 4, 1776* or *American Independence Day*). Aliases can also associate searchable text with pictures or

multimedia elements. Aliases are usually assigned to fields. How-To 11.3 demonstrates the use of aliases.

- ü Define topic groups that the user can select from, to limit the scope of a search operation. These topic groups do not have to be the same groups used for Browse operations, but may include Browse groups if desired. A topic can be included in more than one Searchable group. How-To 11.4 demonstrates the use of Searchable topic groups.
- ü Use {vfld137438953482} fuzzy searches {vfld-9223356093936173056} for words, by comparing only the word stems, rather than the entire words. The selected words are usually, but not necessarily, closely related. For example, a search for *meteorite* would match with terms including *meteor* and *meteoric*. You might also expect it to match *meteorology*, but it doesn't. How-To 11.4 demonstrates the use of fuzzy searches.
- ü Exclude common words from the Search index to reduce the index size and speed up the search. Viewer already eliminates common terms such as *the*, *and*, and *or*—you can add any other words use frequently that would not be useful for searching. How-To 11.4 demonstrates this procedure.
- ü Control the handling of special characters, such as à or æ. This procedure is not demonstrated in this book, but is explained in the Viewer documentation.
- ü Create a custom Search dialog box, which may use specialized search logic. This requires moderately difficult programming. A sample custom Search program is included the MVSAMPLE directory on the enclosed CD-ROM disk.

The response to a search operation is a list of topics that meet the specified criteria. The search can be repeated with additional limits to reduce the number of topics if desired. When the user double-clicks on one of the listed topics, Viewer displays that topic with all of the matching terms highlighted. Figure 11-2 shows an example of this list and a topic with highlighted entries.

```
{ewc vwrht2, TsTextButton, "Figure  
11-2"} [Macro=JI('viewerht.mvb>SecWin', 'fig11_2')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

## Use Multiple Keyword Tables?

Complexity: EASY

### Problem

I want to include an index {vfld137438953482} keyword {vfld4572278980522016768} list, but I have so many keywords that the user may be confused. I need to define two or more separate lists, and let the user choose which list to use.

### Technique

You use Project Editor to define two keyword lists, then use Topic Editor to define keywords within each list.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP11\_1, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP11\_1. Create these 4 directories for all projects, even if some are not needed.
2. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP11\_1 directory. Enter the name of your document file as TEXT\CHAP11\_1.RTF.

Next define the index keyword groups:

3. Choose Keyindex from the Section menu. The Main Keyword Index group appears. Change the Title to Standard Terms.
4. Click on the New button to define a new group, and change the Title to Technical Terms. Click on OK, then save the updated project file.

Next create the document file:

5. Use Project Editor to start Word and create your document, and create a new topic with context string topic\_1. Be sure to delete the page break created before your topic. Create a Topic Title of Introduction.
6. Type the first two paragraphs from the introduction to this chapter into the document. Separate them with a blank line, or use a paragraph format with an extra indentation on the first line, so that you can easily distinguish the paragraphs. Enter about 24 blank lines after the text.
7. Move the insertion point to the beginning of the topic and activate Topic Editor. Select Keywords (K footnote) and click on OK. The Standard Terms keyword group is already selected.
8. Enter the following terms in the Topic Keywords list, then click on OK.  
Enter one term on each line:

large application  
locate  
hierarchy

- Without moving the insertion point, define another Keyword footnote. Select Technical Terms from the Keyword Index pull-down list, then enter the following Topic Keywords:

Index  
Search

- Move the insertion point to the beginning of the second paragraph, and enter the following terms in the Standard Terms keyword group:

index  
categories

- Define a Keyword footnote for the Technical Terms group, as you did in step 9, and enter the terms:

index  
topics  
keywords

- Create a new topic with context string topic\_2 and Title Search Introduction. Copy the third and fourth paragraphs from the introduction to this chapter in the same way as in step 6, including the extra blank lines.

- Create the following keyword footnotes for the first paragraph, in the same way as in steps 7 through 9:

Standard Terms:  
author  
user  
Technical Terms:  
search  
index

- Create the following keyword footnotes for the second paragraph, in the same way as in steps 10 and 11:

Standard Terms:  
author  
criteria  
Technical Terms:  
search  
index

- Save your document, and compile and test as usual. The index dialog box should look like Figure 11–3.

{ewc vwrht2, TsTextButton, "Figure  
11½3"[Macro=JI('viewerht.mvb>SecWin', `fig11\_3')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}

- Select entries defined in the first and second paragraphs of either topic. The appropriate paragraph should appear at the top of the window. Select an entry that shows 2 under the # of topics column of the dialog box. A second dialog box appears, showing the list of topic titles containing the term and allowing you to choose which topic to display.

**How It Works**

You didn't have to define an index group in earlier How-Tos because the main group is always defined by default. When you want to use more than one group you must define them through Project Manager, as you did in steps 3 and 4. After defining the groups, you can add keywords to the new groups through Topic Editor.

**Comment**

The extra blank lines allow the last paragraph to scroll to the top of the window when a keyword defined in that paragraph is selected. If they are not added, the topic scrolls as far as possible, but one or more earlier paragraphs also display.

## Define Search Categories with Fields and Data Types?

Complexity: ADVANCED

### Problem

I want to define several categories of terms to use when searching for topics. These categories include character text, dates, and numbers.

### Technique

You define each category as a field, associate it with the proper type of data, and assign portions of your text to the appropriate field. You use part of the introduction to this chapter as the document text, just as you did in the previous How-To.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP11\_2, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP11\_2. Create these four directories for all projects, even if some are not needed.
2. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP11\_2 directory. Enter the name of your document file as TEXT\CHAP11\_2.RTF.

Next define the fields:

3. Choose Searchdlg from the Section menu to display the Search definition dialog box. (This dialog box, and the one used to perform searches, are both titled Search. To prevent confusion, they are described here as the Search definition and Search operation dialog boxes.) Click on the New button to create a new field, named Field 10
4. Leave the Search Data Type unchanged, enter All Text in the Title field, enter 0 as the Search Field Number, and leave the Word Wheel Subfilename blank. The completed dialog box should look like Figure 11-4.

```
{ewc vwrht2, TsTextButton, "Figure
11-4"}[Macro=JI('viewerht.mvb>SecWin', 'fig11_4')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

5. Click on the New button again. Leave the Search Data Type unchanged, enter Technical Terms in the Title field, leave the Search Field Number unchanged as 10, and enter FLD10 in the Word Wheel Subfilename.
6. Click on the New button again to create Field 11. Choose Number from the Search Data Type pull-down list and enter How-To Sections as the Title. Leave the Search Field Number unchanged and the Word Wheel

Subfilename blank.

7. Click on the New button once again to create Field 12. Choose Date from the Search Data Type pull-down list and enter Dates as the Title. Leave the Search Field Number unchanged and the Word Wheel Subfilename blank. Click on OK, then save your updated project file.

Next create your document file:

8. Use Project Editor to start Word and create your document, and create a new topic with context string `topic_1`. Be sure to delete the page break created before your topic. Create a Topic Title of Topic 1.
9. Type the text of the first four bullets of this chapter's introduction into this topic as separate paragraphs.
10. Create a new topic with context string `topic_2` and title Topic 2. Type the text of the remaining four bullets from the introduction into this topic.
11. Select the word *search* in the first paragraph of the first topic. Activate Topic Editor and select Search field.
12. Two elements, both named Search Field—All Text, appear. Leave the first one selected, and select Technical Terms from the Search Field pull-down list on the right side of the dialog box.
13. Click on OK. Your text should now contain `{vfld10}search{vfld}`. The second `vfld` entry, which was produced by the second Search Field element in the dialog box, resets the field number to its default value of zero. This allows the text that follows the word you selected to be included in the All Text category.
14. Repeat the process in steps 11 through 13 to define all occurrences in both topics of the words *search*, *index*, *browse*, and *format* as Technical Terms fields.
15. Select the How-To section number (11.2) at the end of the first paragraph. Activate Topic Editor and select Search field with data type.
16. Four elements appear—two Search fields and two data types. Use the pull-down lists to select How-To Sections for the first search field element, and Number for the first data type element. Leave the second element of each type unchanged. Click on OK.
17. Repeat the process in steps 15 and 16 to define the date in the third paragraph of the first topic, formatted as 07/04/1776, as a Dates search field with a data type of Date. Your document should look like Figure 11-5.

```
{ewc vwrht2, TsTextButton, "Figure  
11½5"[Macro=JI('viewerht.mvb>SecWin', `fig11_5')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

18. Save your document, and compile and test as usual.
19. Click on the Search button to display the basic Search operation dialog box. Click on the Search by Category button to enlarge the dialog box by including the Search by Category section. Notice that the button you just clicked has been recaptioned as Search by Word. Clicking on that



removes the Category section.

20. Select Technical Terms from the Category pull-down list. Notice that a pull-down list becomes available to select a keyword, as shown in Figure 11-6. Select search from that list, then click on OK.

```
{ewc vwrht2, TsTextButton, "Figure  
11i½6"[Macro=JI('viewerht.mvb>SecWin', `fig11_6')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

21. You see the first topic, with all occurrences of the word *search* highlighted, plus a dialog box that lets you choose which topic you want to see, as shown in Figure 11-7.

```
{ewc vwrht2, TsTextButton, "Figure  
11i½7"[Macro=JI('viewerht.mvb>SecWin', `fig11_7')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

22. Try the other categories. They don't have keyword lists, so you have to enter terms you know exist, as explained below. What happens if you use *search* as a keyword in the All Text category? Use the Add Row button to create compound searches, looking for topics with more than one selected keyword in one or more categories.

### How It Works

Viewer creates the {vfld137438953482} All Text {vfld-9078975914968088576} category (field 0) automatically if you have any topics with titles (\$ footnotes). This category includes all of the text in those topics, including text that is part of other categories. You defined the category because it wouldn't appear in the Category pull-down list otherwise; it would only be used in the Search by Word alternative to searching by category. When this category is defined, the user can use either method the same way, and can combine terms from the general and specific categories in a single compound search operation.

A {vfld137438953482} pull-down list {vfld-9223356093936173056} of keywords, such as you found for the Technical Terms category, is produced as a result of placing an entry in the Word Wheel Subfilename field when the category is defined. The Viewer documentation refers to this as a *word wheel list*. If you leave that field blank, the user must enter a term without assistance. Viewer won't create such a list for field 0. That's just as well—it would be too big, since that field contains all of your text.

### Comment

The {vfld137438953484} field numbers {vfld4221583484780544} you can use start with 10. Fields 0 through 9 are reserved by Viewer. Field 0 is the All Text category and field 1 is text in the non-scrolling region. The other fields have not been identified.

Viewer does not support defining fields within fields. As a side-effect of this, defining fields in the {vfld137438953484}non-scrolling region {vfld3413442438365184} (all of which is field 1) is officially unsupported. It works in some cases and not in others, but cannot be counted on. You can, however, achieve the same result through the use of aliases, as demonstrated in How-To 11.3.

The Options button in the Search dialog box offers the user the chance to restrict searching to “{vfld137438953484}topic titles {vfld1045116047360786432} only.” This is an inaccurate description—it really means to search only the non-scrolling regions. This is why the text in those regions is in field 1.

The Search definition dialog box, shown in Figure 11-4, allows you to define the meaning of the term {vfld2305858952132296714}*near*{vfld-9223357193447800832} as it is used when the user performs a search. Viewer lets the user search for topics where one specified term is within a certain number of words of a second specified term, as *term1 near term2*. You can specify the default number of words in the definition dialog box. The user can choose a different value through the dialog box displayed when the user clicks on the Options button in the Search operation dialog box.

Only topics that have titles (\$ footnotes) are included in the Search operation. This is why the compiler reports {vfld2305858952132296716}*Warning 6176*{vfld-9223357193447800832}, *No words in titled topics found to index; MVB file will lack full-text index* for most of the How-Tos in this book—none of the topics have titles.

The search operation always displays the selected topics in the master pane of the main window. Topics that are not normally displayed there, such as popups, are usually created without titles to exclude them from the search. How-To 11.3 shows how to work around this if you want to include selected terms from such topics in your search functions.

The numeric data types require entries in acceptable formats, as follows:

{vfld137438953482}Numbers {vfld-9223356093936173056} (data type 1) can contain a minus sign, commas, and decimal points. Scientific notation, which includes the letter *E* followed by an exponent, can also be used.

{vfld137438953482}Dates {vfld-9223356093936173056} (data type 2) must be either in the form *mm/dd/yyyy*, or a four-digit year with an optional suffix of *B* to indicate a BC date. The year can be up to 11,767,033 (AD or BC).

{vfld137438953482}Times {vfld-9223356093936173056} (data type 3) use the form *hh:mm:ss.dd*. PM times use 24-hour times (i.e., 2:00 PM is 14:00). Elapsed times, which may exceed 24 hours, are allowed. The seconds and hundredths are optional.

{vfld137438953482}Epochs {vfld-9223356093936173056} (data type 4) are used for time values that are too large to represent as dates, such as “450 million BC” The format is a number, with optional commas and an optional suffix of *B* to indicate BC.



## Use Fields with Aliases?

Complexity: ADVANCED

### Problem

I want to let users search for dates without using the unattractive date format in my text, and I want to let them search for common synonyms of certain words. I also want to define fields within a non-scrolling region, but this caused many very strange compiler error messages when I tried. The messages all said {vfld2305858952132296716} *Warning 7131{vfld-9079242005371944960}: Field string too long in RTF source.*

### Technique

You define aliases with the desired searchable text in the necessary format. This technique satisfies all three requirements.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

### Steps

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP11\_3, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP11\_3. Create these four directories for all projects, even if some are not needed.
2. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP11\_3 directory. Enter the name of your document file as TEXT\CHAP11\_3.RTF.

Next define the fields and the alias file:

3. Choose Searchdlg from the Section menu to display the Search definition dialog box. (This dialog box, and the one used to perform searches, are both titled Search. To prevent confusion, they are described here as the Search definition and Search operation dialog boxes.) Click on the New button to create a new field, named Field 10.
4. Leave the Search Data Type unchanged, enter All Text in the Title field, enter 0 as the Search Field Number, and leave the Word Wheel Subfilename blank.
5. Click on the New button again to create Field 11. Leave the Search Data Type unchanged and enter Technical Terms as the Title. Leave the Search Field Number unchanged and enter wwtech as the Word Wheel Subfilename.
6. Click on the New button once again to create Field 12. Choose Date from the Search Data Type pull-down list and enter Dates as the Title. Leave the Search Field Number unchanged and the Word Wheel Subfilename blank. Click on OK.
7. Choose Alias File from the Section menu to display a standard File Open dialog box. Enter alias113.txt as the file name and click on OK. A

dialog box appears saying that the file does not exist and asking if it should be created. Click on OK to display the Alias File dialog box shown in Figure 11–8. Click on OK, then save your updated project file. You'll add alias definitions later.

```
{ewc vwrht2, TsTextButton, "Figure  
11i½8"[Macro=JI('viewerht.mvb>SecWin', `fig11_8')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Next create your document file:

8. Use Project Editor to start Word and create your document, and create a new topic with context string contents. Be sure to delete the page break created before your topic. Define a topic title of Contents Topic.
9. Without moving the insertion point, choose Paragraph from Word's Format menu and select the Keep With Next check box. Click on OK.
10. Type the first bulleted item from the introduction to this chapter into the topic.
11. Press **[ENTER]** to end the paragraph, then repeat the procedure in step 9 to unselect the Keep With Next option.
12. Insert a blank line to separate the paragraphs, then type the second and third bulleted items from the introduction to this chapter into the topic.

Now define the aliases:

13. Choose the first word in the non–scrolling region to define as a field—use the second word in the text, *search*. Decide which field and data type applies to this alias. This word uses field 11 (Technical Terms), data type 0 (Words).
14. Switch to Project Editor to define the alias by holding **[ALT]** and pressing **[TAB]** until Project Editor is named, then release both keys. Select Alias File from the Section menu to display the Alias File dialog box.
15. Click on the New button to define the first alias, number 0. Set the Search Data Type to 0 (Words), and the Search Field Number to 11 (Technical Terms) by using the pull–down lists. Enter *search* as the Alias Text and click on OK.
16. Repeat step 15 to define alias number 1, but enter *find* as the Alias Text.
17. The Word document window should still be visible on your screen. If it is, click on it to switch back to Word. Otherwise, use **[ALT]–[TAB]** to switch to that application as you did in step 14.
18. Select the word in the non–scrolling region to be aliased, activate Topic Editor, and choose the Search alias entry to display the alias definition dialog box shown in Figure 11–9. Enter 0 as the Alias Number, then click on OK. Repeat the procedure to assign alias number 1 to the same word

```
{ewc vwrht2, TsTextButton, "Figure  
11i½9"[Macro=JI('viewerht.mvb>SecWin', `fig11_9')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

19. Repeat steps 13 through 15 to define the word *index*, near the end of the first sentence, as alias number 2. Enter *index* as the alias text in Project Editor. Assign this alias to the word in the non-scrolling region as you did in step 18.
20. Define aliases for all remaining occurrences of the words *search* and *index* in the document as you did in step 18, using aliases 0 and 1 for *search*, and 2 for *index*.
21. Repeat steps 13 through 15 to define alias number 3 as a data type 2 (date) and field number 12 (Dates) with alias text *7/4/1776*.
22. Select the words *July 4, 1776* in the third paragraph; activate Topic Editor and select the Search alias entry. Enter 3 as the Alias Number, then click on the element name on the left side of the dialog box. This updates the dialog box fields with the definitions from alias number 3, so that you can confirm that you selected the correct alias.
23. Position the insertion point on the line following the last paragraph, then create a new topic with context string *dummy\_topic*.
24. Enter the words *search*, *find*, and *index*. Select each, one at a time, and use Topic Editor's Search field entry to define each word as field 11 (Technical Terms). Note that you use the Search field and data type entry if you want a data type other than 0 (Words).
25. Save your document, and compile and test as usual.
26. Click on the Search button, then Search by Category. Select Technical Terms from the Category pull-down list, then select Find from the Keyword pull-down list. Click on OK to display the results, as shown in Figure 11-10. Notice that the occurrences of the word *search* in both the non-scrolling region and regular text are highlighted. Also notice that the second topic is not referenced.

```

{ewc vwrht2, TsTextButton, "Figure
11i½10"[Macro=JI(`viewerht.mvb>SecWin', `fig11_10')]
[Font="Arial" /S12/B4] /W100 /H40/B1/D2}

```

27. Search in the Dates category for *7/4/1776*. The phrase *July 4, 1776* is highlighted. Try searching for just the year 1776—the same text is found.

### How It Works

In steps 3 through 7 you define the search fields you need. Field number 0 (All Text) is defined so that the user can use both the full text and selected fields in compound searches. This is one of only two supported uses of fields 0 to 9, which are reserved by Viewer. (The other use is in aliases) In step 7 you create an alias file.

It is a little awkward to create aliases until you get used to it. First you use Project Editor to define the alias (as in steps 14 and 15), then you use Topic Editor to assign document text to a defined alias (as in step 17). The alias definition contains four simple pieces of information—the alias number, data

type, field number, and alias text. Alias numbers start with 0 and continue consecutively. You cannot skip any numbers. You can use reserved field numbers (such as 0 for All Text) in an alias definition. The alias text is matched against the user's request in the search operation.

Assigning an alias through Topic Editor is very simple. You select the text you want highlighted in case of a match, select Topic Editor's Search alias entry, and enter the desired alias number. The other fields in this dialog box are filled in from the alias definition. They can be seen if you click on the element field on the left side of the dialog box, as in step 22. You can assign a single alias entry to as many words or phrases in the RTF file as you want, as long as the alias definition and text apply to each. The dialog box you use to define aliases has an option to search for a suitable alias entry.

The easiest way to create aliases is to size and position your windows so you can see both Word and Project Editor at the same time—either side by side or one above the other. The alternative is to switch between them by using Windows' [ALT]–[TAB] task switching feature.

You define aliases in the non-scrolling region (NSR) so that text in that region can be included in a search field. The compiler errors occur because the entire region is automatically part of reserved field 1. Viewer does not support field definitions within field definitions, so defining fields within NSR text usually doesn't work. It may work in some cases, but the results are unpredictable. It usually causes the poorly worded Warning 7131.

You create the {vfld137438953482} dummy topic {vfld-9042384167995703296} in steps 23 and 24 because alias text is not included in the list of keywords (word wheel list). You solve this problem by creating a dummy topic containing definitions of the desired values for each field. This topic does not have a title, which prevents these entries from being selected by the search operation. The topic is not included in a Browse sequence or referenced in a hot spot, which prevents users from displaying this topic.

### **Comment**

The most common use of aliases is to allow searching on numeric information, including dates and times, while using a more pleasing form of the information in the topic text. For example, *July 4, 1776* is a better form for your text than *7/4/1776*. Aliases can also define alternate names, such as *Samuel Clemens* for *Mark Twain*, and define fields within NSRs.

Aliases can associate searchable terms with pictures by placing the alias command in the text immediately before the command that displays the picture and specifying 0 characters to be highlighted. The easiest way to do this is to place the insertion point where you want the alias command, then activate Topic Editor without selecting any text. Then choose the Search alias entry and enter an alias number as usual. The alias itself is defined as usual, with the term to be used for searching.

Several aliases can be defined for a single term by repeating the standard alias definition and assignment procedure as required. You end up with a series of alias commands in your document, preceding the aliased text.

The search operation always displays the selected topics in the master pane of the main window. If you want to let the user search text that is usually displayed in a popup or other specialized window, the results may not be attractive. If the specialized window is displayed by a hot spot, you can

solve this by defining the desired terms as aliases to that hot spot. If the window is displayed by a topic entry command, define appropriate aliases in the first paragraph of the topic containing the topic entry command. When that topic is displayed by the search operation, the command executes, causing the other window to also be displayed. These aliases usually do not highlight any text.

{vfld2305857852620668940} Alias {vfld13331578486784}ed terms can't be used with the {vfld2305858952132296716}*near*{vfld-9223357193447800832} operator in search operations. This is by design, and Microsoft does not consider it to be a bug. Aliased terms do work with the Boolean search operators, such as *and* and *or*.



**Limit and Simplify Searches?**

Complexity: EASY

**Problem**

I want to simplify and speed up searches by eliminating words I use often, letting the user select which groups of topics should be searched, and accepting words that are similar, not just identical, to the ones requested.

**Technique**

Viewer already excludes many common words. This How-To excludes others by adding them to the stop file. Searchable topic groups are defined to let the user select the topics to be searched. A different word-breaker routine is selected to implement

{vfld137438953482}fuzzy searches {vfld4572278980522016768} based on the stems of the requested words.

You create the standard directories, project file, and document file for this How-To. (To review those procedures, refer to sections 3.1 and 3.2.)

**Steps**

First prepare the directories and files:

1. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP11\_4, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP11\_4. Create these four directories for all projects, even if some are not needed.
2. Use the File Manager to copy USA.STP from the MVTOOLS subdirectory of your Viewer Publishing Toolkit directory (usually MVPUBKIT) into your project directory, VIEWERHT\CHAP11\_4.

Next update the project file:

3. Use Viewer's Project Editor to create a new project file in your VIEWERHT\CHAP11\_4 directory. Enter the name of your document file as TEXT\CHAP11\_4.RTF.
4. Choose FTINDEX from the Section menu to display the Data Types dialog box.
5. Leave the Data Type Number as 0, and change the DLL Routine Name to {vfld2305865549202063371}FBreakAndStemWords{vfld-9079242005371944960}. This is the routine that provides fuzzy searches. Click on the ellipsis (...) after the Stop-Word List Filename field, then select file USA.STP. The completed dialog box should look like Figure 11-11. Click on OK.

```
{ewc vwrht2, TsTextButton, "Figure
11½"}[Macro=JI('viewerht.mvb>SecWin', `fig11_11')]
[Font="Arial" /S12/B4] /W100 /H40/B1/D2}
```

6. Choose Groups from the Section menu, and click on New to create a

group named Group1. Clear the Searchable check box for this group. Create Group2, select the Searchable check box, and enter Second Group as the Title. Create Group3 with the check box selected and title Third Group. Click on OK.

7. Save your updated project file.

Now update the {vfld137438953482}stop file{vfld-35321252696555520} to exclude selected words:

8. Start Notepad, and open file USA.STP from your project directory, VIEWERHT\CHAP11\_4. You see about three dozen numbers and words, listed in alphabetic order, as shown in Figure 11–12.

```
{ewc vwrht2, TsTextButton, "Figure  
11_12"}[Macro=JI(\viewerht.mvb>SecWin', `fig11_12')]  
[Font="Arial" /S12/B4] /W100 /H40/B1/D2}
```

9. Insert the following terms, in the proper alphabetic sequence: can, How-To, and Viewer. Save the file and exit Notepad.

Now create your document file:

10. Use Project Editor to start Word and create your document, and create a new topic with context string topic\_1 and title First Topic. Be sure to delete the page break that is created before your topic.
11. Activate Topic Editor and select Topic Groups (+ footnote). Use the pull-down list to select Group1 in the Browse Sequence field, and enter 010 as the Browse Sequence Number. Use the pull-down list for the Topic Groups field to select the Group2 entry. Click on OK.
12. Type the first bullet from the introduction to this chapter into this topic.
13. Create a new topic with context string topic\_2, and title Second Topic.
14. Activate Topic Editor and select Topic Groups again. The Browse Sequence and Browse Sequence Number fields still contain the values used in the first topic. Leave the Browse Sequence unchanged, and change the Browse Sequence Number to 020.
15. Use the Topic Groups pull-down list to select Group2. Click on the Insert Group button to create a blank line in the list of groups, then select Group3 from the pull-down list to add that group to the list of Topic Groups, as shown in Figure 11–13. Click on OK.

```
{ewc vwrht2, TsTextButton, "Figure  
11_13"}[Macro=JI(\viewerht.mvb>SecWin', `fig11_13')]  
[Font="Arial" /S12/B4] /W100 /H40/B1/D2}
```

16. Copy the second bullet from the introduction to this chapter into this topic.
17. Create a new topic with context string topic\_3 and title Third Topic. Activate Topic Editor and select Topic Groups again. Change the Browse

Sequence Number to 030. Select Group3 in the Topic Groups list. Click on OK, then copy the fifth bullet (which discusses fuzzy searches) from the introduction into this topic.

18. Save the document file and compile and test as usual.
19. Try to search for **Viewer**—you get a message box saying, *Your query is empty or has no searchable words* because you are searching for a term in the stop file.
20. Try searching for meteor or meteorite. See which words match each one.
21. Try limiting your searches to the second or third topic groups. The second group finds words in the first or second topics, and the third group finds words in the second or third topics. Notice that the first group is not listed in the dialog box.

### **How It Works**

In steps 4 and 5 you select a different DLL routine to process words while searching your file. The new routine comes with Viewer. It extracts the stem of each word, in both the document and your search request, and matches on those shorter forms of the words. This results in more matches, but includes many selections you don't want. It can be difficult to predict which words will or won't match when using this type of comparison.

In step 5 you also assign a stop file, USA.STP, for words in your document. You copy the default file into your project directory in step 2, and update it in steps 8 and 9. This file contains a list of terms to be excluded from the search operation, which makes the file index smaller and speeds up the search. Note that no stop file is used, not even the default, unless you request it through this dialog box.

You create three topic groups in step 6. Only the second and third are searchable, so the first does not appear in the Search dialog box. You create three topics in steps 10 through 17, and assign all of them to one Browse group. You also assign the first two topics to Group2, and the second and third topics to Group3.

### **Comment**

Each topic can be in only one Browse Sequence (or none), but can be in many Topic Groups. The same group definitions commonly apply to both browsing and search limiting, because the same logical grouping may be reasonable for both purposes.

Viewer only includes a fuzzy-search DLL routine to process English words. Routines for other languages can be written, if you can decide how to determine the roots of the words in the desired languages. One author has already developed such a routine for `{vfld137438953482}French{vfld-35322352208183296}` words. If you are interested in this routine, refer to Appendix D.

Always include a stop file in any application that permits searching. Even the standard file reduces the size of the index greatly.



## 11.5 Tips and Tricks

- ⇒ Always design searching and indexing capabilities in the context of your topic structure and other navigational aids. Your goal should always be to make all of these features operate logically, consistently and easily.
- ⇒ If your application has so much information that you think it needs both searchable topic groups and multiple field definitions, you should seriously consider writing a custom search dialog box that makes it easy to select the desired options—otherwise, your users could be overwhelmed by the variety of options.
- ⇒ Aliases can be useful when your users may be used to certain terms that are synonyms for terms you use. For example, anyone who uses word processing programs frequently might think of *find*, rather than *search*. By making *find* an alias for *search*, as in How-To 11.3, this operation could become easier for some users.
- ⇒ Use Word's Find menu choice to locate occurrences of terms you want to define as fields or aliases. The term remains highlighted when you exit the dialog box, so it's ready for you to activate Topic Editor. You can also use Word's Replace menu choice to find the terms and insert the desired field or alias command in one fast and automatic operation. Just enter the desired term in the Find What field, and the sequence of command and term in the Replace With field.
- ⇒ The standard search function lists topics ordered by the hit density—the ratio of matched words to total words in the topic. For example, suppose a search returns two topics and the first has three matched words out of 50, and the second has five matched words out of 200. The first topic will be listed first because it has a higher density, even though it has fewer matched words.

**Search**

Search by Word

Search In:

All Topic Groups  
 Selected Topic Groups  
 Current Topic Only

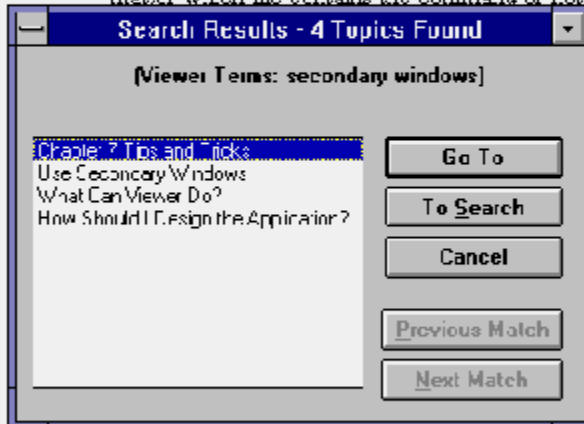
**Topic Groups:**

- Chapter 1: What Is Viewer?
- Chapter 2: Designing a Viewer Applic
- Chapter 3: Creating a Simple Applicat
- Chapter 4: Adding Graphics

Search by Category

| Category:    | Keyword:             |
|--------------|----------------------|
| Viewer Terms | <input type="text"/> |
|              | browse groups        |
|              | browse sequence      |
|              | Bullet               |
|              | Button               |
|              | button bar           |
|              | caption              |
|              | categories           |
|              | CDAudio              |

⇒ If you want to use **secondary windows** in conjunction with multiple files, you must be careful to define the secondary window in the file containing the topics displayed in that window. It doesn't matter which file contains the command or hot spot that causes the



to solve them by part of a special support group entry or How-To 7.2. The presence of group and group, the group commands. If the file, it may be directory. You can get the project by double-clicking on using the standard Windows sequence—the current directory, the Windows directory, the Windows System directory, and the directories listed in the DOS PATH command in the AUTOEXEC.BAT file. If the file still isn't found, Viewer checks the [FILES] section of the

**Index**

Type a word, or select one from the list.

| Index list        | # of Topics |
|-------------------|-------------|
| <b>author</b>     | <b>2</b>    |
| categories        | 1           |
| criteria          | 1           |
| hierarchy         | 1           |
| index             | 1           |
| large application | 1           |
| locate            | 1           |
| user              | 1           |

Current index:



[SEARCHDLG] - Search Dialog

**Search Categories:**

All Text

**Search Data Type:**  
0 - Words

**Title:**  
All Text

**Search Field Number:**  
0

**Word Wheel Subfilename:**  
|

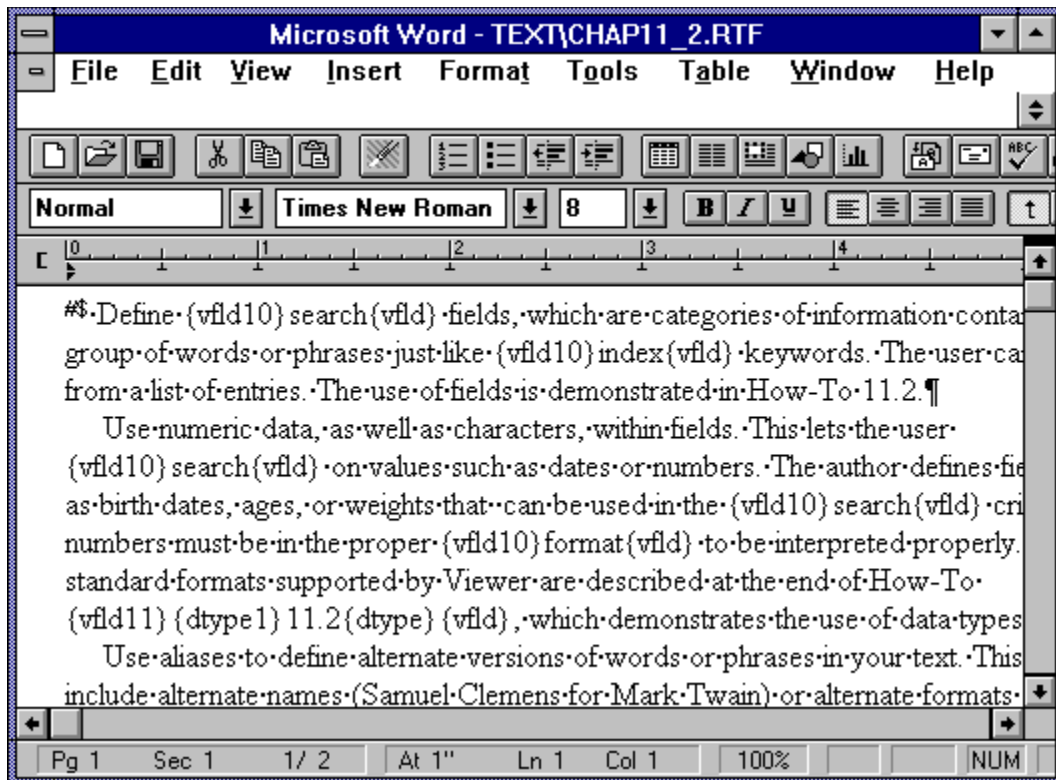
**Comment:**

Delete New

OK  
Cancel

Help

NEAR means within 8 words.



**Search**

Search by Word

Search In:

All Topics

Current Topic Only

List of Previous Topics Found

OK

Cancel

Options...

Hints...

<< Search by Word

Search by Category

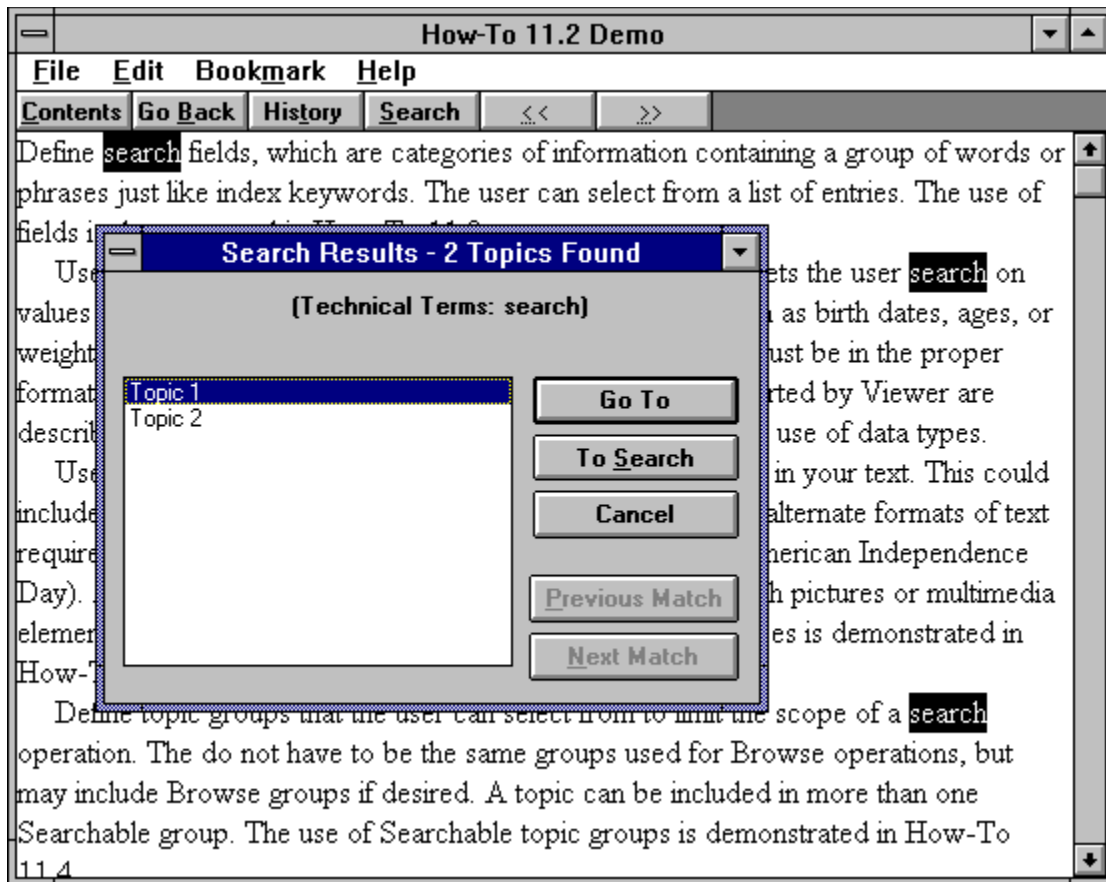
| Category:       | Keyword: |
|-----------------|----------|
| Technical Terms | search   |

Browse  
format  
index  
search

Add Row

Remove Row

Previous Search





Viewer Topic Editor - CHAP11\_3.MVP

**Viewer Elements:**

Search alias 6 Characters

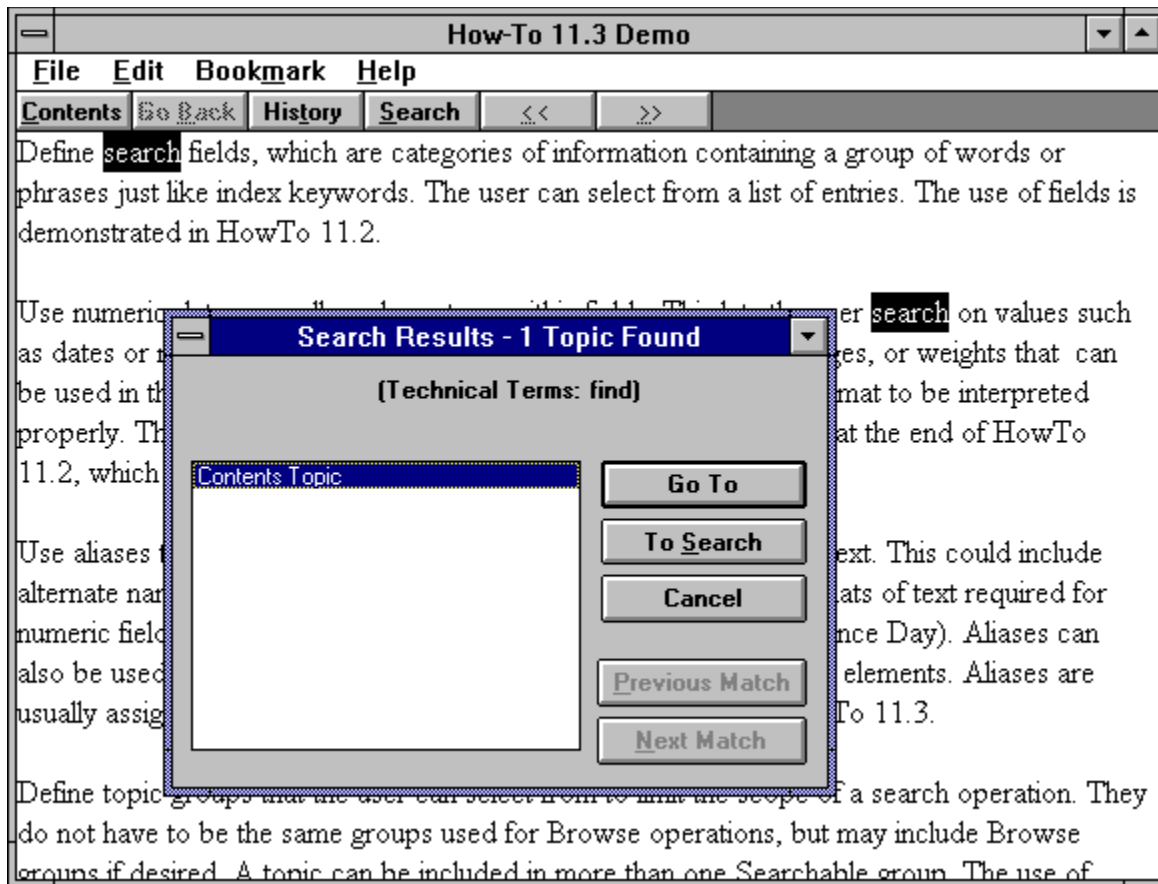
Alias  characters in RTF file as:

Alias **N**umber:

Search Data Type:

Search Field:

Alias Text:



[FTINDEX] - Data Types

**Search Data Types:**

- 0 - Stemmed Words
- 1 - Number
- 2 - Date
- 3 - Time
- 4 - Epoch

**Data Type Number:**  
0

**DLL Filename:**  
mvbrkr2.dll

**DLL Routine Name:**  
FBreakAndStemWords

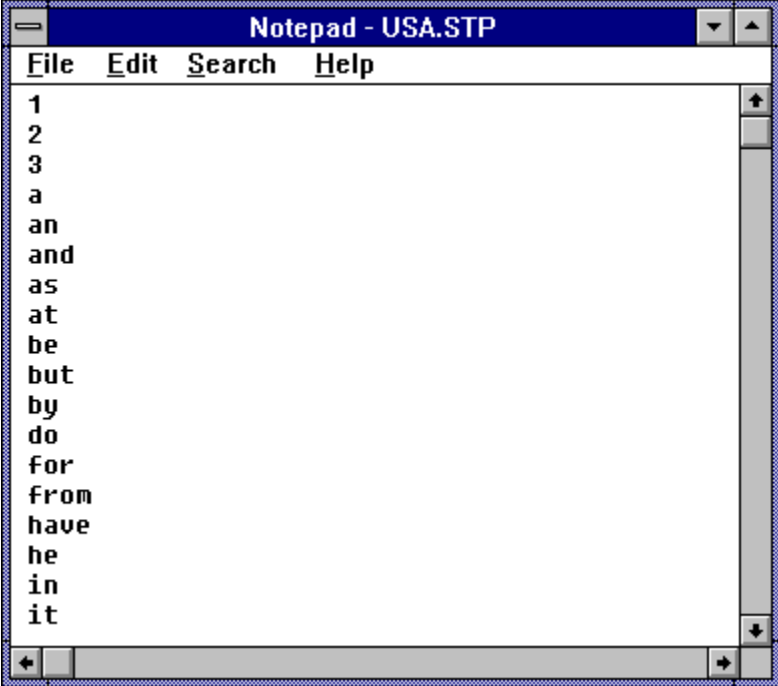
**Stop-Word List Filename:**  
usa.stp

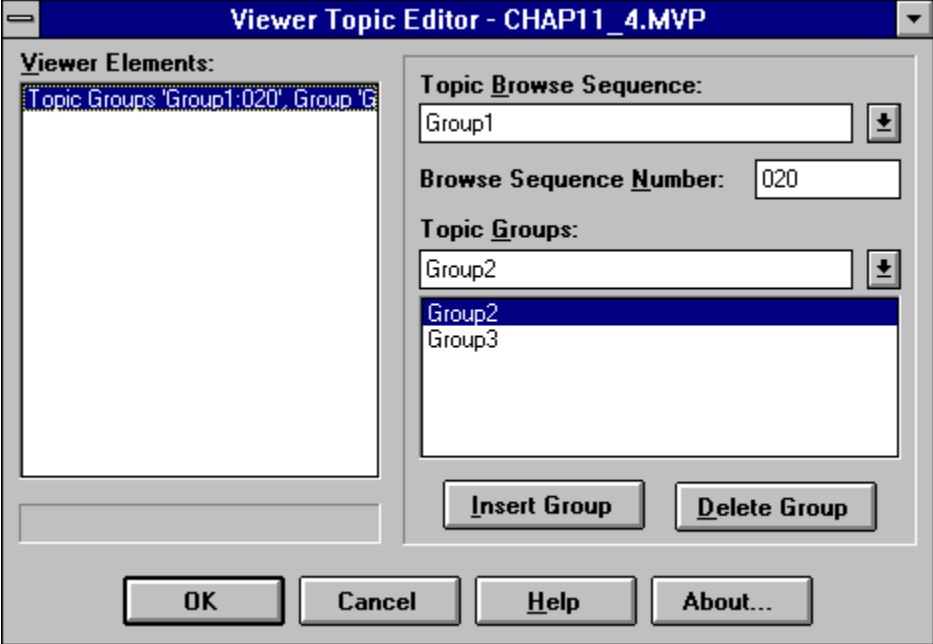
**Character Tables Filename:**

**Comment:**  
reakAndStemWords to use word stemming

**Delete** **New** **OK** **Cancel** **Help**









In the previous chapters you worked with individual features in artificial demonstrations. Now it's time to take those features, and other Viewer capabilities, and produce some real-world examples. This chapter creates two common types of Viewer applications—a reference application and a training application—that use techniques that apply to many types of Viewer applications. In this chapter you create small-scale demonstration or pilot versions of these applications. These demonstrations are not intended to be complete. The testing steps and Comment sections discuss issues related to designing full-scale applications. This approach helps you understand and appreciate the importance of design decisions.

The reference application is an encyclopedia, although most of the techniques serve just as well for other references. Reference applications can cover a wide range of subjects (as in an encyclopedia), or concentrate on a single subject such as photography, Windows APIs, or music. Microsoft's initial Viewer applications are all reference applications—Bookshelf is a set of reference books, Encarta is an encyclopedia, Cinemania concentrates on movies, and the Developer Network CDs provide technical material for developers. Viewer is popular for developing reference applications because of the powerful tools that are provided for locating information—hot spots and the other hypertext navigation aids, the keyword index, and the full-text and author-controlled search functions.

Viewer is also well suited to developing training materials, especially for use within a company. Besides the information-locating tools used in reference applications, training materials can make excellent use of pictures with multiple hot spots (great for letting the user choose part of a form or CRT screen to study) and animations. Sounds and movies can also be valuable ways to make certain points. For example, these can demonstrate dealing with customers or performing physical operations that are difficult to explain clearly using only text.

As you work on these applications, remember that the techniques you are using are also appropriate for many other types of applications. For example, an auto repair manual could let the user click on parts of the auto in a picture to see a closer view, until a particular part is selected. Hot spots or buttons could offer suggestions for testing, removal, part numbers, and replacement.

After doing these How-Tos, experiment by adding other features, and create similar demo or pilot versions of other applications. Remember that the compiler included with this book limits you to 25 topics in your document file. Have fun!'

## Create a Reference Application?

Complexity: ADVANCED

### Problem

I want to create an encyclopedia. Users should be able to locate topics from an alphabetic list, by category, and by searching for desired words or phrases. The topics should include pictures, sounds, and movies where appropriate and available. It should start by offering instructions on using the application for first-time users.

### Technique

The encyclopedia contains a selection of topics in several categories, based on the picture and movie files available. The Search function provides the primary method for locating subjects, but a two-level alphabetic menu of topics is also included. A complete help file is provided, and a chance to read those instructions is offered when the file is loaded. A new button makes it easier to copy topic contents to other applications. Note that the files used by this demonstration require over 36 megabytes of disk space! Most of this is due to the movie files.

This How-To uses many techniques demonstrated in previous sections. If you want to review the procedures, refer to the indicated sections:

- ü Create the standard directories, project file, and document file (How-Tos 3.1 and 3.2).
- ü Use a graphics hot spot (How-To 4.4).
- ü Display 256-color pictures (How-To 4.6).
- ü Create a greeting topic (How-To 5.10).
- ü Create a button (How-To 6.1).
- ü Create a custom About window (How-To 6.3).
- ü Use multiple files (How-To 7.1).
- ü Play sounds (How-Tos 8.1–8.6).
- ü Play AVI movies (How-To 9.1).
- ü Use fields in searches (How-To 11.2).
- ü Use groups to limit searches (How-To 11.4).

### Steps

First, plan the topics, context strings, topic groups, fields, aliases, and other design details:

1. Select the topics that provide the lowest level of detail. Descriptions of all the planned topics, context strings, and files for each topic are shown in Table 12-1.
2. Design the supporting topics you know are required. In this demo these are an initial menu, three submenus (A-K, L-P, Q-Z), the initial greeting, and a custom About screen. These topics are also shown in Table 12-1. Popup topics are designed as they are needed.

**Table 12-1. Planned Topics for How-To 12.1**

| Category         | Title              | Context String | Files    |
|------------------|--------------------|----------------|----------|
| American History | John F. Kennedy    | ctx_JFK        |          |
| kennedy.avi      |                    |                |          |
|                  | Martin Luther King | ctx_MLK        |          |
| mlkingjr.avi     |                    |                |          |
|                  | Theodore Roosevelt | ctx_TR         |          |
| roosvelt.avi     |                    |                |          |
| Animals          | Duck               | ctx_duck       |          |
| ducks.avi        |                    |                |          |
|                  | Elephant           | ctx_elephant   |          |
| elephant.avi     |                    |                |          |
|                  | Horse              | ctx_horse      |          |
| horses.avi       |                    |                |          |
| Space            | Saturn             | ctx_saturn     |          |
| saturn.bmp & wav |                    |                |          |
|                  | Shuttle            | ctx_shuttle    |          |
| liftoff.avi      |                    |                |          |
|                  | Sun                | ctx_sun        |          |
| solar.bmp & wav  |                    |                |          |
| Music            | Guitar             | ctx_guitar     |          |
| guitar.mid       |                    |                |          |
|                  | Piano              | ctx_piano      |          |
| piano.mid        |                    |                |          |
|                  | Ragtime            | ctx_ragtime    |          |
| ragtime.mid      |                    |                |          |
|                  | Rock Music         | ctx_rock       | rock.mid |
| Support          | Initial Menu       | contents       |          |
|                  | Submenu 1          | menu_ak        |          |
|                  | Submenu 2          | menu_lp        |          |
|                  | Submenu 3          | menu_qz        |          |
|                  | Initial Greeting   | ctx_greet      |          |
|                  | About Screen       | ctx_about      |          |

3. A topic group is needed for each of the categories of detail topics (American History, Animals, Music, and Space). These groups are used for both the Browse and Search operations.
4. One field, containing topic titles, is required. The standard All Text field is always created if any other text fields are used. No aliases or other special features are required.

Next, prepare the directories and files:

5. Use the Windows File Manager to create your project directory, VIEWERHT\CHAP12\_1, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP12\_1. You create these four directories for all projects, even if some are not needed.
6. Use the File Manager to copy the picture, sound, and movie files from the appropriate VIEWERHT\HOWTOS\CHAP12\CHAP12\_1 subdirectories on the CD-ROM to the corresponding VIEWERHT\

CHAP12\_1 subdirectories on your hard drive. The wave files are in the CHAP12\_1 directory, not the SOUNDS subdirectory, and must be copied to the corresponding directory on your hard drive. The commands used to play these files do not support paths—the files must be in the same directory as the MVB file.

7. Use the File Manager to copy VWRHELP.MVB from the MVHLPUSA subdirectory of your Viewer directory (usually MVPUBKIT) to your project directory, VIEWERHT\CHAP12\_1. Copy the USA.STP file similarly, from the MVTOOLS subdirectory.
8. Use Project Editor to create a new project file in your VIEWERHT\CHAP12\_1 directory. Enter the name of your document file as TEXT\CHAP12\_1.RTF.

Now enter the necessary definitions in the project file:

9. Choose Window Definitions from the Section menu, then click on the Properties button. Enter How-To 12.1 as the Window Caption. Use the System Control menu box to close the Window Properties dialog box, then click on OK.
10. Choose Title Options from the Section Menu, and enter contents in the Contents Topic field, and Copyright (C) 1993 your name in the Copyright field. Click on OK.
11. Choose Config from the Section menu, and paste in an AppendItem command. Select mnu\_help from the pull-down list for the MenuID field, change the NewItemID to `mnu\_about\_demo', change the ItemCaption to `About &How-To 12.1', and change the Command to `{vfld137438953483}PopuID{vfld-9079242005371944960}(qchPath,"ctx\_about)". Click on OK.
12. Paste in another AppendItem command, and select mnu\_help as the MenuID again. Change NewItemID to `mnu\_help', ItemCaption to `Using &Viewer', and Command to `JumpID("vwrhelp.mvb","HelpMain)". Click on OK.
13. Paste in an InsertButton command, and enter `btn\_copy' as the ButtonID and `&Copy' as the button caption. Select `CopyTopic()' from the pull-down list for the Command field, and enter 6 as the button position. Click on OK.
14. Paste a JumpID command into the end of the Config script. Edit the command, and enter qchPath in the TitleFile field, one space in the WindowName field, and `ctx\_greet' (with single quotes) in the Context field. Click on OK in each dialog box. Figure 12-1 shows the complete configuration script.

```
{ewc vwrht2, TsTextButton, "Figure
12½1"[Macro=JI('viewerht.mvb>SecWin', `fig12_1')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

15. Choose FTINDEX from the Section menu, select the Words search data type, and choose USE.STP as the stop word file. Click on OK.
16. Select Groups from the Section menu, then click on New to create the

- first group. Change the name to AmHist and the title to American History. Leave the Searchable checkbox selected.
17. Click on New to create another group, and change the name to Animals and the title to Animals. Click on New again, and enter Space in the name and title fields. Click on New once again, and enter Music in the name and title fields. Click on OK.
  18. Choose Searchdlg from the Section menu, and click on the New button to create a new field. Change the Title field to All Text, and the Search Field Number to 0.
  19. Click on New again, and enter Subjects as the Title, and ww10 as the Word Wheel Subfilename. Leave all other fields unchanged and click on OK.
  20. Save the updated project file.

Next create the support topics:

21. Use Project Editor to start Word and create your document, and create a new topic with context string contents and title Table of Contents. Be sure to delete the page break that is created before your topic.
22. Enter the following text using the Times New Roman font, 20 point size, and boldface:  
This is the main menu. Click on the range of subjects below that contains the subject you want:
23. Insert a few blank lines, then create a hot spot with the text Aardvark—King that jumps to context string menu\_ak. Continue to use the same character format as in step 21 throughout this topic.
24. Insert a blank line, then create a similar hot spot with text Label—Puzzle and context string menu\_lp.
25. Insert a blank line, then create a similar hot spot with text Quack—Zulu and context string menu\_qz.
26. Create a new topic with context string menu\_ak and title Menu A—K.
27. Enter the following text in 16–point Times New Roman, centered:  
**CLICK ON THE DESIRED SUBJECT:**
28. Insert a couple of blank lines, then three columns of topics as follows. Continue using 16–point Times New Roman, with paragraph format option Keep Lines Together. (This prevents word wrapping.) Use tabs to line up the columns.

|            |          |                     |
|------------|----------|---------------------|
| Aardvark   | Doodle   | Guitar              |
| Abdicate   | Duck     | Habitat             |
| Advertise  | Ear      | Horse               |
| Badge      | Elephant | Human               |
| Benefactor | Error    | Incurable           |
| Blunder    | Feather  | Ivory               |
| Cat        | Flash    | Jab                 |
| Compensate | Funny    | Kennedy, John F.    |
| Curtail    | Gab      | Kennedy, Robert     |
| Defect     | Greek    | King, Martin Luther |

29. Select the word *duck*, and create a hot spot that jumps to ctx\_duck.



Create similar hot spots so that *Elephant* jumps to `ctx_elephant`, *Guitar* jumps to `ctx_guitar`, *Horse* jumps to `ctx_horse`, *John F. Kennedy* jumps to `ctx_JFK`, and *Martin Luther King* jumps to `ctx_MLK`.

30. Restore the normal paragraph formatting, and insert a couple of blank lines. Create a centered hot spot with text Return to Main Menu that jumps to contents. This topic should look like Figure 12–2.

```
{ewc vwrht2, TsTextButton, "Figure
12i½2"[Macro=JI('viewerht.mvb>SecWin', 'fig12_2')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

31. Create a new topic with context string `menu_lp` and title Menu L—P.
32. Use the same formatting and heading line as in steps 26 and 27 and enter the following list of topics:

|         |          |          |
|---------|----------|----------|
| Label   | Move     | Offer    |
| Let     | Mud      | Opposite |
| Lift    | Navigate | Ordeal   |
| Loop    | Neat     | Overcome |
| Love    | Nice     | Pact     |
| Luck    | Noose    | Passage  |
| Man     | Novice   | Petite   |
| Mammoth | Numb     | Piano    |
| Misuse  | Oar      | Pluck    |
| Moon    | Obstruct | Puzzle   |

33. Select *Piano* and create a hot spot that jumps to `ctx_piano`.
34. Restore the normal paragraph formatting, and insert a couple of blank lines. Create a centered hot spot with text Return to Main Menu that jumps to contents.
35. Create a new topic with context string `menu_qz` and title Menu Q—Z.
36. Use the same formatting and heading line as in steps 26 and 27 and enter the following list of topics:

|              |           |            |
|--------------|-----------|------------|
| Quack        | Secede    | Umbr       |
| Quarrel      | Shuttle   | Understand |
| Quest        | Speculate | Utopia     |
| Quit         | Sun       | Vacant     |
| Ragtime      | Tact      | Vegetable  |
| Reaping      | Tendon    | Waffle     |
| Rock Music   | Thief     | Wrench     |
| Roosevelt, T | Tontine   | X-Ray      |
| Ruler        | Trademark | Yelp       |
| Saturn       | Ultimate  | Zulu       |

37. Make hot spots such that *Ragtime* jumps to `ctx_ragtime`, *Rock Music* jumps to `ctx_rock`, *T Roosevelt* jumps to `ctx_TR`, *Saturn* jumps to `ctx_saturn`, *Shuttle* jumps to `ctx_shuttle`, and *Sun* jumps to `ctx_sun`.
38. Restore the normal paragraph formatting, and insert a couple of blank lines. Create a centered hot spot with text Return to Main Menu that jumps to contents.
39. Create a new topic with context string `ctx_greet` and title Greeting. Enter the text Welcome to the Demo Encyclopedia.
40. Insert a few blank lines, then create a hot spot with text Click here to read instructions for using this application, and jumping to context

string HelpMain in the VWRHELP.MVB file.

41. Insert a few blank lines, then create a hot spot with text Click here to display the Main Menu, and jumping to context string contents.
42. Create a new topic with context string ctx\_about and no title. Enter the following text:

Demo Encyclopedia, created in  
How-To 12.1 in the book  
Microsoft Multimedia Viewer How-To  
by Stephen Pruitt

Next create the topics that use AVI movies:

43. Create a new topic with context string ctx\_JFK and title John F. Kennedy. Create a Browse group footnote for the AmHist group. Leave the sequence number blank for the sake of simplicity. Enter the topic title (John F. Kennedy) as a heading line, using 14-point type, and define that line as a search field (Subjects). Insert a few blank lines, then insert a Viewer multimedia command that specifies MCI device AVIVideo, file name MOVIES\KENNEDY.AVI, position Left, and caption Watch a movie. All other options should remain in the default values. Insert a couple of blank lines, then enter the following text:

John F. Kennedy was the 35th President of the United States. He was inaugurated in January 1961 and was assassinated in Dallas, Texas on November 22, 1963. He was the first Roman Catholic to be elected President. The Berlin Wall was built during his administration, in 1961.

This topic should look like Figure 12-3 in your document.

```
{ewc vwrht2, TsTextButton, "Figure  
12i½3"[Macro=JI('viewerht.mvb>SecWin', 'fig12_3')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

44. Create a topic similar to the previous with context string ctx\_MLK, title and heading Martin Luther King, and file name MOVIES\MLKINGJR.AVI. Define the heading as a Subjects search field. Use the same movie caption as in step 43. Create a Browse group footnote for the AmHist group without a sequence number. Enter the following text:

Martin Luther King was a prominent leader of the American civil rights movement until his assassination on April 4, 1968. His philosophy stressed non-violence and the dignity and common humanity of all people. He was the leader of voter registration drives throughout the South, and the civil rights march on Washington in 1963. He delivered his famous "I Have a Dream" speech at a rally in Washington during that march. He received the Nobel Peace Prize in 1964.

45. Create a topic similar to the previous one with context string ctx\_TR, title and heading Theodore Roosevelt, and file name MOVIES\ROOSVELT.AVI. Define the heading as a Subjects search field. Use the same movie caption as in step 43. Create a Browse group footnote for the AmHist group without a sequence number. Enter the following text:

Theodore Roosevelt was the 26th President of the United States,

from 1901 until 1909. He expanded the powers of the federal government in conflicts between Big Business and Big Labor by advocating the concept of a public interest. He used executive powers to bring suits against a number of business trusts. He also had legislation passed to control railroad rates and begin inspection of food and drugs. Under his leadership, vast expanses of federal lands were set aside for conservation.

46. Create a topic similar to the previous one with context string `ctx_duck`, title and heading `Duck`, and file name `MOVIES\DUCKS.AVI`. Define the heading as a Subjects search field. Use the same movie caption as in step 43. Create a Browse group footnote for the Animals group without a sequence number. Enter the following text:

Ducks are waterfowl in the family *Anatidae*, which includes geese and swans. Ducks are found throughout the world. They are identified by their webbed feet and flattened bills.

47. Create a topic similar to the previous one with context string `ctx_elephant`, title and heading `Elephant`, and file name `MOVIES\ELEPHANT.AVI`. Define the heading as a Subjects search field. Use the same movie caption as in step 43. Create a Browse group footnote for the Animals group without a sequence number. Enter the following text:

Elephants are large mammals that live in Africa, India, and southern Asia. They have been domesticated and trained to carry heavy loads. They live in herds and are highly intelligent. The Asian elephant has smaller ears and a smaller body than its African counterpart. Female Asian elephants do not have tusks.

48. Create a topic similar to the previous one with context string `ctx_horse`, title and heading `Horse`, and file name `MOVIES\HORSE.AVI`. Define the heading as a Subjects search field. Use the same movie caption as in step 43. Create a browse group footnote for the Animals group without a sequence number. Enter the following text:

Horses are a hoofed mammal belonging to the family *Equidae*. They have a long association with man and have been used for transport, hauling, and sport. Selection and breeding have resulted in a variety of types, each suited to a particular function.

49. Create a topic similar to the previous one with context string `ctx_shuttle`, title and heading `Shuttle`, and file name `MOVIES\LIFTOFF.AVI`. Define the heading as a Subjects search field. Use the same movie caption as in step 43. Create a Browse group footnote for the Space group without a sequence number. Enter the following text:

The shuttle is America's first reusable spacecraft. It is used for both government and private commercial projects. Examples of shuttle missions include:  
March, 1989 - *Discovery* launched a Data Relay satellite and tested a model of a heated pipe radiator for the planned space station.  
May, 1989 - *Atlantis* launched the Magellan radar mapper to Venus.  
October, 1989 - *Atlantis* launched the Galileo space probe that travelled to Jupiter by way of Venus.

Next create the topics that use bitmaps and wave sound files:

50. Create a new topic with context string `ctx_saturn` and title `Saturn`. Create a Browse group footnote for the Space group, without a sequence number. Enter the title text as a heading line, using 14-point type, and

define that line as a search field (Subjects). Insert a few blank lines, then insert a Viewer ewX picture command. Select the Any Display file folder tab, then choose file PICTURES\SATURN.BMP, store file in Baggage, dither picture on 16-color displays, position Left, and type the centered caption Click on picture to play music. Paste in an MCICommand command, edit the MCI parameters to `saturn.wav', and select Play from the MCI Command pull-down list. All other options should remain in the default values. Insert a couple of blank lines, then enter the following text:

Saturn is the sixth planet from the sun. It has a diameter of 72,000 miles and revolves around the sun once every 29.5 years. It is famous for the series of thin, flat rings that circle it.

51. Repeat the procedure in step 50 to create a topic about the Sun. Enter `ctx_sun` as the context string, and Sun as title and heading. Select file PICTURES\SOLAR.BMP, and play the `solar.wav' sound file. All other options are the same as in step 50. Enter the following text:

The Sun is a star, just like most of the tiny points of light you see in the sky. The picture shows a phenomenon called solar prominences clearly. The sun is the center of our solar system, providing light and heat to the planets.

Next create the topics that use MIDI sound files:

52. Create a new topic with context string `ctx_guitar` and title Guitar. Create a Browse group footnote for the Music group, without a sequence number. Enter the title text as a heading line, using 14-point type, and define that line as a search field (Subjects). Insert a few blank lines, then insert a Viewer multimedia command that specifies MCI device sequencer, file name SOUNDS\GUITAR.MID, position Left, and caption Play music. Use a custom controller containing only the Play/Pause button and no slider. All other options should remain in the default values. Insert a couple of blank lines, then enter the following text:

The guitar is a stringed instrument played by plucking. The long finger board, or neck, of the guitar has a series of small raised strips of metal called frets. The guitar is based on an invention 5,000 years ago in Egypt during the days of the pharaohs. The rise in popularity of rock music in the 1950s and 1960s subjected many households to the sounds of a teenage would-be guitar superstar.

53. Create a similar topic with context string `ctx_piano`, title and heading Piano, and playing the SOUNDS\PIANO.MID file. Define each heading as a Subjects search field. All other options should be the same as in step 52. Enter the following text:

The piano is a keyboard instrument, based on a device invented by Bartolommeo Cristofori, the curator of musical instruments for the Medici family in Florence, Italy. Its popularity results from its great versatility, including the range of octaves and variations in volume from very soft to very loud. In addition, it lends itself to the expression of many musical styles, including classical, jazz, ragtime, blues, and rock and roll.

54. Create a similar topic with context string `ctx_ragtime`, title and heading Ragtime, and playing the SOUNDS\RAGTIME.MID file. Define each heading as a Subjects search field. All other options should be the same

as in step 52. Enter the following text:

Ragtime is a style of piano jazz that was very popular in the early 1900s. The music employs a strongly accented syncopated pattern over a strong steady left hand beat. The rhythm is jerky or “ragged,” hence the name. The basic rhythm makes frequent use of a pattern of sixteenth-eighth-sixteenth notes. The most famous ragtime pianist and composer was Scott Joplin (1869-1917), a black pianist from Texas.

55. Create a similar topic with context string `ctx_rock`, title and heading Rock Music, and playing the `SOUNDS\ROCK.MID` file. Define each heading as a Subjects search field. All other options should be the same as in step 52. Enter the following text:

Rock has been a popular musical style since the 1950s. It is commonly played using the guitar, electronic synthesizer, or piano. It is characterized by a strong, driving beat. There are many styles of rock music, from the early “bubble gum” to the later heavy metal style.

Now add some inter-topic jumps and popups:

56. Make the words *guitar* and *piano* in the Ragtime and Rock Music topics hot spots that jump to context strings `ctx_guitar` and `ctx_piano`. Make the terms *Ragtime* and *Rock* in the Guitar and Piano topics hot spots that jump to context strings `ctx_ragtime` and `ctx_rock`.

57. Locate terms in the detail topics that would be unfamiliar to some users. Use *civil rights movement* (under Martin Luther King), *prominences* (under Sun), and *frets* (under Guitar). Make each term a popup hot spot with a suitable context string. Create topics with those context strings, no titles, and the following text:

Civil rights movement:

The American civil rights movement had as its goal securing equal rights for black Americans. It worked to eliminate forced segregation, voting rights abuses, and other forms of discrimination.

Prominences:

Solar prominences are bursts of solar material that shoot out from the surface, as a result of uneven heating and combustion within the sun.

Frets:

Frets mark off intervals of half steps, helping the player to find the correct pitch. The fingers press down on the strings just above the frets.

Figure 12–4 shows a section of the document file.

```
{ewc vwrht2, TsTextButton, "Figure  
12i½4"[Macro=JI('viewerht.mvb>SecWin', `fig12_4')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

Now finish up and try it out:

58. Save the document file as usual, and save the project file again to make sure that all changes are saved, then compile and test the application.
59. Try jumping to the help file, shown in Figure 12–5. Would a user know

how to return to the main application?

```
{ewc vwrht2, TsTextButton, "Figure  
12i½5"[Macro=JI('viewerht.mvb>SecWin', `fig12_5')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

60. Go to each of the detail topics through the menus. Would a user know how to use the multimedia controller buttons? Did you figure out how to stop the music in the Saturn and Sun topics? (The solution is to leave the topic.) Would a user figure that out? Figure 12–6 shows the topic about the Sun.

```
{ewc vwrht2, TsTextButton, "Figure  
12i½6"[Macro=JI('viewerht.mvb>SecWin', `fig12_6')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

61. Use the browse buttons to move within general topic categories. How would a user know what these buttons do?
62. Use the Search operation, and search by category. Select the Subjects category, and look at the pull–down list of subjects shown in Figure 12–7. Do you really need both this and the initial menus? What other categories would be useful?

```
{ewc vwrht2, TsTextButton, "Figure  
12i½7"[Macro=JI('viewerht.mvb>SecWin', `fig12_7')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

In steps 1 through 4 you design the basic components of the application. This makes the authoring process easier and helps you avoid forgetting anything important. In steps 9 and 10 you define the window caption, copyright information, and contents topic.

Steps 11 and 12 add entries to the standard Help menu. One displays a custom About window and the other displays the Viewer help information from a separate MVB file. The topic displayed in this window is created in step 42.

Step 13 defines a new button that copies the current topic to the Windows Clipboard.

Step 14 causes a greeting topic to be displayed when the application is loaded.

Step 15 applies the standard stop word file, to eliminate common words from the search index.

The topic groups are defined in steps 16 and 17. These groups are used for browsing and to limit search operations.

Steps 18 and 19 define the search categories (also known as fields).

The main menu and submenus are created in steps 21 through 38. These are simple jump hot spots arranged in columns. The greeting topic is created

in steps 39 through 41. This uses hot spots to allow the user to jump to the help file or to the main menu. This topic is never displayed again during one execution of the application.

The topics that use AVI movies are created in steps 43 through 49. The movie caption helps to tell the user what the controller buttons do. All of the subject topics have browse footnotes without sequence numbers. This causes the topics within each group to be displayed in their physical order in the file when the browse buttons are used.

Steps 50 and 51 create topics that have 256-color pictures and sound files. Dithering is used to display the pictures on 16-color systems best. If the user clicks on the picture, a command is executed to play a sound file.

Steps 52 through 55 create topics that play MIDI sound files using the Viewer multimedia embedded window command. A controller containing only a Play/Pause button is used to show the effect of a minimal controller.

Jumps between related topics are added in step 56, and popup hot spots are added in step 57.

### **Comment**

The source files used to create the Viewer help file are included with Viewer and on the CD-ROM disk enclosed with this book. They can be modified to reflect the techniques included in your application. You can compile them into your MVB file if desired.

Group entry and exit command scripts can be used to simplify the use of the help topics, including returning to the main application. Viewer commands could be executed when entering or leaving the help topics to change and restore the contents topic, for example. A button could be added, while in the help topics, that jumps to the application's main menu. The help topics could be displayed in a secondary window by using techniques demonstrated in How-Tos 5.6 and 6.9.

The `{vfld137438953482}browse sequence{vfld280933810831360}` number was left blank in each topic. This causes the topics to be displayed in their physical order in the file. A real reference would use sequence numbers that display the topics in alphabetic order or some other meaningful sequence.

A short summary of the user interface would be helpful. This would be offered in the initial greeting and the Help menu. Using a consistent interface in all subject topics makes it much easier to explain the functions to the user.

**Create a Training Package?**

Complexity: ADVANCED

**Problem**

I want to create a training package. It must be very easy to use, requiring minimal instructions. The users must have very little control over what topics are displayed. It should include pictures, sounds, and animations where appropriate.

**Technique**

This training package teaches new employees how to answer the telephone as a switchboard operator, receptionist, or department secretary. There are four sections—Introduction, Basics, Holding Calls, and Taking Messages. It includes a custom interface, displayed in a regular pane alongside the training material, that replaces the standard Viewer menus and buttons. Users are offered a brief description of how to use the training package when the file is loaded. The standard Search, Browse, History, and other navigational aids are not used.

A standard message pad is shown, with each portion selectable as a hot spot that displays instructions for filling in that part of the form.

This How-To uses many techniques that were demonstrated in previous sections. Refer to the following sections to review the procedures:

- ü Create the standard directories, project file, and document file (How-Tos 3.1 and 3.2).
- ü Use multiple hot spots in a picture (How-To 4.5).
- ü Use a table to display information (How-To 5.9).
- ü Display a greeting topic (How-To 5.10).
- ü Use a graphic interface in a regular pane (How-Tos 6.6 and 6.8).
- ü Play sounds automatically (How-To 8.3).
- ü Play a spoken greeting (How-To 8.4).
- ü Play an animation (How-To 9.6).

**Steps**

First, plan the topics, context strings, and other design details:

1. Select the topics that provide the training material, including the popups used for the message pad. Descriptions of all the planned topics with their context strings are shown in Table 12-2.
2. Design the supporting topics you know are required. In this demo these are an initial greeting and the instructions. Four topics contain the custom interface, each reflecting a different current section of the course. These topics are also shown in Table 12-2.

**Table 12-2. Planned Topics for How-To 12.2**

| Description  | Context String |
|--------------|----------------|
| Introduction | ctx_intro      |



|                           |                  |
|---------------------------|------------------|
| Basics                    | ctx_basics       |
| Holding Calls             | ctx_hold         |
| Taking Messages           | ctx_message      |
| Message Pad: For          | ctx_pad_for      |
| Message Pad: Date/Time    | ctx_pad_when     |
| Message Pad: Caller       | ctx_pad_caller   |
| Message Pad: Of           | ctx_pad_of       |
| Message Pad: Phone Number | ctx_pad_phone    |
| Message Pad: Message      | ctx_pad_message  |
| Message Pad: Signed       | ctx_pad_signed   |
| Message Pad: Urgent       | ctx_pad_urgent   |
| Message Pad: Checkboxes   | ctx_pad_boxes    |
| Greeting                  | ctx_greeting     |
| Instructions              | ctx_instructions |
| Controls—all              | ctx_control_all  |
| Controls #1               | ctx_control_1    |
| Controls #2               | ctx_control_2    |
| Controls #3               | ctx_control_3    |
| Controls #4               | ctx_control_4    |

---

- No topic groups are needed because the browse and search operations are not used in this demonstration. No fields, aliases, or topic titles are needed for the same reason.

Next, prepare the directories and files:

- Use the Windows File Manager to create your project directory, VIEWERHT\CHAP12\_2, and the standard subdirectories, TEXT, SOUNDS, PICTURES, and MOVIES, under CHAP12\_2. Create these four directories for all projects, even if some are not needed.
- Use the File Manager to copy the picture, sound, and movie files for this How-To from the appropriate HOWTOS\CHAP12\_2 subdirectories on the CD-ROM to the corresponding VIEWERHT subdirectories on your hard drive. The wave files are in the CHAP12\_1 directory, not the SOUNDS subdirectory, and must be copied to the corresponding directory on your hard drive. The commands used to play these files do not support paths—the files must be in the same directory as the MVB file.
- Use the Segmented Hot Spot Editor to create the multiple hot spot file MSGPAD.SHG from MSGPAD.BMP in the PICTURES subdirectory. Mark each section of the picture corresponding with entries in Table 12-2 that begin with Message Pad, and define each section as a popup hot spot that pops up the context string listed. Figure 12-8 shows the check box section, in the middle of the form, being defined as a hot spot that pops up context string ctx\_pad\_boxes.

```
{ewc vwrht2, TsTextButton, "Figure
12i_1/28"[Macro=JI( viewerht.mvb>SecWin', `fig12_8')] [Font="Arial"
```

/S12/B4] /W100 /H40/B1/D2}

7. Use Project Editor to create a new project file in your VIEWERHT\CHAP12\_2 directory. Enter the name of your document file as TEXT\CHAP12\_2.RTF.

Now enter the necessary definitions in the project file:

8. Choose Window Definitions from the Section menu, then click on the Properties button. Enter How-To 12.2 Demo as the Window Caption, and set the Use Default Color check box. Click on the Master Pane button, then clear the Auto-Position check box to allow the master pane to be resized. Enter 225 in the Left field, and 798 in the Max Width. Use the System Control menu box to close the Window Properties dialog box.
9. Click on the Panes file folder tab, then click on New to create a regular pane named Panel. Click on the Properties button, and change the pane name to Controls and select (none) for the Border. Click on the Windows button to display the Pane Associations dialog box, select the Show in Window check box, then click on the Preview button to display the window layout. Drag the edges of the Controls pane to fill the left side of the window, then use the System Control menu box to close the preview window. Figure 12-9 shows the pane layout. Close the Window properties dialog box the same way, then click on OK to close the Window Definitions dialog box.

{ewc vwrht2, TsTextButton, "Figure  
12i½9"[Macro=JI('viewerht.mvb>SecWin', 'fig12\_9')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}

10. Choose Title Options from the Section Menu, and enter ctx\_greeting in the Contents Topic field. Leave the Copyright field blank because the About box can't be displayed in this application. Select PICTURES\CHAP12\_2.ICO as the icon file. Click on OK.
11. Choose Config from the Section menu. Delete the Std20Menus() and Std20Buttons() commands by highlighting them and pressing [DEL].
12. Paste in a RegisterRoutine command, and enter `mmsystem' as the DLLName, `waveOutGetNumDevs' as the function name, and `u=' as the parameter spec.
13. Paste in another RegisterRoutine command, and enter `mmsystem' as the DLLName, `sndPlaySound' as the function name, and `Su' as the parameter spec.
14. Paste in another RegisterRoutine command, and enter `user' as the DLLName, `MessageBox' as the function name, and `USSu' as the parameter spec. Figure 12-10 shows the completed configuration script. Click on OK to return to the main Project Editor window, then save the updated project file.

{ewc vwrht2, TsTextButton, "Figure  
12i½10"[Macro=JI('viewerht.mvb>SecWin', 'fig12\_10')] [Font="Arial"

/S12/B4] /W100 /H40/B1/D2}

Now create the supporting topics:

15. Use Project Editor to start Word and create your document, and create a new topic with context string `ctx_greeting`. Be sure to delete the page break that is created before your topic.
16. Create a topic entry footnote, and type in the following command, all on one line:

```
IfThenElse(waveOutGetNumDevs(), "sndPlaySound(`badans.wav', 1",  
"{vfld137438953483}MessageBox{vfld-9078975914968088576}  
(hwndApp, `Your system does not support sound', `ERROR', 0)")
```

17. Click on OK to return to the document. Enter the text **Answering the Telephone**, using 16-point Arial type centered in the top line of the topic.
18. Insert a few blank lines, then create a table with one row and two columns. Select the first column, then choose **Column Width** from the **Table** menu. Set the column width to 4 inches, and the space between to 1 inch. Click on the **Next Column** button, and set the width of the second column to 2 inches.
19. Enter the following text in the left cell of the table:

What did you think?

Hard to understand? If you were calling, how would you feel about a company that has someone answering the phone that way? Do you want to talk with this person? Would you trust a person who gives this impression to take your message properly?

This course teaches you how to answer the phone so that you make a good impression on the caller, and handle calls effectively.

20. In the right cell insert a **Viewer ewX** picture command to display file **PICTURES\DESKTOP.BMP**. Use character position and no caption. Check the **Store Picture in Baggage** check box.
21. Insert a few blank lines, then enter the text **CLICK TO HEAR A PROPER ANSWER** using 14-point type, and centered. Make this text a hot spot and paste in an **MCICommand**. Enter `!correct.wav` as the MCI parameters and `play` as the MCI command. Click on OK in each dialog box to return to the document.
22. Insert a couple of blank lines, then create two text hot spots on one line. The first says **CLICK FOR INSTRUCTIONS**, and jumps to context string `ctx_instructions`. The second says **CLICK TO PROCEED**, and jumps to context string `ctx_intro`. The completed topic in the document is shown in Figure 12-11.

```
{ewc vwrht2, TsTextButton, "Figure  
12i½11"[Macro=JI(`viewerht.mvb>SecWin', `fig12_11')]  
[Font="Arial" /S12/B4] /W100 /H40/B1/D2}
```

23. Create a new topic with context string `ctx_instructions`. Create a topic entry **PaneID** command and enter `qchPath` as the title file, one space as

the window name, `ctx_control_all` as the context string, Controls as the pane name, and 0 as the value for `PrintTabCopyOrder`.

24. Enter the text How to Use This Course, using 16–point Arial type centered in the top line of the topic.
25. Insert a few blank lines, then insert the topic text from a file by choosing File from Word’s Insert menu, then selecting the `CHAP12_2\INSTRUCT.DOC` file.

Now create the topics containing the custom interface controls:

26. Create a new topic with context string `ctx_control_all`. Insert a Viewer `ewX` picture command, specify `PICTURES\INTRON.BMP` as the file, and select the Store Picture in Baggage check box. Paste in a `JumpID` command and enter `qchPath` as the file name, one space for the window name, and `ctx_intro` for the context string. Click on OK.
27. Insert one blank line, then insert a Viewer `ewX` picture command. Specify file `PICTURES\BASICSN.BMP` and Store Picture in Baggage. Paste in a `JumpID` command and enter `qchPath` as the file name, one space for the window name, and `ctx_basics` for the context string. Click on OK.
28. Insert one blank line, then insert a Viewer `ewX` picture command. Specify file `PICTURES\HOLDN.BMP` and Store Picture in Baggage. Paste in a `JumpID` command and enter `qchPath` as the file name, one space for the window name, and `ctx_hold` for the context string. Click on OK.
29. Insert one blank line, then insert a Viewer `ewX` picture command. Specify file `PICTURES\MSGN.BMP` and Store Picture in Baggage. Paste in a `JumpID` command and enter `qchPath` as the file name, one space for the window name, and `ctx_message` for the context string. Click on OK.
30. Insert one blank line, then insert a Viewer `ewX` picture command. Specify file `PICTURES\EXITSML.BMP` and Store Picture in Baggage. Paste in an Exit command. This does not take any parameters, and does not need to be edited. Click on OK. The completed topic is shown in Figure 12–12.

```
{ewc vwrht2, TsTextButton, "Figure  
12½12"[Macro=JI('viewerht.mvb>SecWin', `fig12_12')] [Font="Arial"  
/S12/B4] /W100 /H40/B1/D2}
```

31. Create a new topic with context string `ctx_control_1`. Repeat steps 26 through 30 except use file `INTROA.BMP` in the first picture.
32. Create a new topic with context string `ctx_control_2`. Repeat steps 26 through 30 except use file `BASICSA.BMP` in the second picture.
33. Create a new topic with context string `ctx_control_3`. Repeat steps 26 through 30 except use file `HOLDA.BMP` in the third picture.
34. Create a new topic with context string `ctx_control_4`. Repeat steps 26 through 30 except use file `MSGGA.BMP` in the fourth picture.

Now create the training topics:

35. Create a new topic with context string `ctx_intro`. Create a topic entry footnote and paste in a `PanelID` command. Enter `qchPath` (without quotes) as the title file, one space for the window name, `ctx_control_1` as the context string, `Controls` as the name of the pane, and `0` as the `PrintTabCopyOrder` value
36. Enter the text `You Are the Company!`, using 16–point Arial type, centered in the top line of the topic.
37. Insert a couple of blank lines, then insert a `Viewer` multimedia command. Enter `GDAnim` as the MCI device, select the `MOVIES\COMPANY.AWA` file, select the text aligned position check box, and select the `Looping` and `Auto–start playback options` check boxes. Clear the `Show Controller` checkbox. Center the command text so that the resulting command will be centered.
38. Insert a couple of blank lines, then the following text:  
When a customer calls for the first time, your answer forms their first impression of the company. You are the company! Think about how you react to the way different companies answer the phone.

Try to be...

COURTEOUS – be polite, and never argue  
CHEERFUL – a friendly voice is always pleasant  
CONFIDENT – convince callers you'll handle the request properly  
CLEAR – be sure the caller understands you

39. Create a new topic with context string `ctx_basics`. Create a topic entry footnote and paste in a `PanelID` command. Enter `qchPath` (without quotes) as the title file, one space for the window name, `ctx_control_2` as the context string, `Controls` as the name of the pane, and `0` as the `PrintTabCopyOrder` value.
40. Enter the text `Basic Concepts`, using 16–point Arial type, centered in the top line of the topic.
41. Insert the text for this topic into your document from a separate file by choosing `File` from `Word's Insert` menu, then selecting the `CHAP12_2\BASICS.DOC` file.
42. Create a new topic with context string `ctx_hold`. Create a topic entry footnote and paste in a `PanelID` command. Enter `qchPath` (without quotes) as the title file, one space for the window name, `ctx_control_3` as the context string, `Controls` as the name of the pane, and `0` as the `PrintTabCopyOrder` value
43. Enter the text `The Call on Hold`, in 16–point Arial type, centered in the top line of the topic.
44. Insert a few blank lines, then the following text:  
If you put a caller on hold while waiting for the desired person to become available, keep checking to see if the caller still wants to hold. Think how you would feel if you decided to give up and leave a message, but no one came back to give you that chance.

You should also offer to transfer the caller to someone else who

might be able to assist him or her.

45. Create a new topic with context string `ctx_message`. Create a topic entry footnote and paste in a `PaneID` command. Enter `qchPath` (without quotes) as the title file, one space for the window name, `ctx_control_4` as the context string, `Controls` as the name of the pane, and `0` as the `PrintTabCopyOrder` value

46. Enter the text `Taking a Message`, in 16–point Arial type, centered in the top line of the topic.

47. Insert the following text:

Fill out a “While You Were Out” message form for every call where the caller leaves a message or identifies himself or herself.

The standard form is shown below. Click on each portion of the form to get an explanation of that section.

48. Insert a Viewer `ewX` picture command with file `PICTURES\MSGPAD.SHG`. Select the `Store Picture in Baggage` check box and text aligned position. Center the command text so that the resulting command will be centered.

Next create the message pad popup topics:

49. Create a new topic with context string `ctx_pad_for`, and enter the following text:

Enter the name of the person this message is for. Make sure it’s clear, so you aren’t wondering later “Which Don was that for?”

50. Create a new topic with context string `ctx_pad_when`, and enter the following text:

Enter the date and time you received the call.

51. Create a new topic with context string `ctx_pad_caller`, and enter the following text:

Enter the name of the person on the phone here. Make *sure* you have the right spelling!

52. Create a new topic with context string `ctx_pad_of`, and enter the following text:

Enter the name of the company the caller represents.

53. Create a new topic with context string `ctx_pad_phone`, and enter the following text:

This is the number to call back to reach the person on the phone. Try to get a number even if the caller says “He knows it.”

54. Create a new topic with context string `ctx_pad_message`, and enter the following text:

Ask the person on the phone if he or she wants to leave a message. If so, write the message here. Make sure it’s clear, and make sure of any unfamiliar words.

55. Create a new topic with context string `ctx_pad_signed`, and enter the following text:  
Enter your name here. If the person getting this message has any questions, this lets him or her know who took the message.
56. Create a new topic with context string `ctx_pad_urgent`, and enter the following text:  
Put a checkmark in this box if the caller says this message is urgent. Make sure you get this message to the right person as quickly as possible.
57. Create a new topic with context string `ctx_pad_boxes`, and enter the following text:  
Place checkmarks where appropriate. Is the caller expecting a call back, or will he or she call again? Was this call returning a previous call?
58. Save the document and project files, and compile and test the application. The first course section is shown in Figure 12–13.

```
{ewc vwrht2, TsTextButton, "Figure
12i½13"[Macro=JI('viewerht.mvb>SecWin', 'fig12_13')] [Font="Arial"
/S12/B4] /W100 /H40/B1/D2}
```

### How It Works

In the first three steps you design the application. This includes deciding what topics are needed, choosing a context string for each topic, and designing the user interface. The supporting topics, topic groups, fields, and other components are determined at the same time, based on the design decisions.

A multiple hot spot picture is created in step 6 that is used later, in one of the application's topics.

In steps 8 and 9 the master pane is resized and a regular pane for the user interface is created.

The standard menus and buttons are eliminated in step 11, and external routines are defined in steps 12 through 14.

The greeting topic is created in steps 15 through 22. Step 16 causes a sound file to be played automatically if the computer system supports playing sounds, and displays an error message otherwise. Steps 18 through 20 use a table to arrange text and a picture in the middle of the topic. Step 21 creates a text hot spot that causes a sound file to be played.

A topic containing a short set of instructions is created in steps 22 through 25. Text from a separate file is inserted into the topic in step 25.

Steps 26 through 30 create a topic containing the custom interface displayed with the instructions, and steps 31 through 34 create the interface topics displayed with the four sections of this course. The interface uses a picture hot spot for each subject. A distinctive picture is displayed for the active subject, and the other three subjects are represented by standard pictures. Each of these pictures are hot spots that jump to the appropriate

topic. A hot spot picture is also included to exit the application.

The topic containing the first subject is created in steps 35 through 38. A topic entry command is defined in step 35 that displays the proper custom interface topic in the regular pane, and a multimedia controller that plays an animation is created in step 37. This displays the animation automatically, and keeps repeating it as long as this topic is displayed. There aren't any control buttons, so it can't be stopped.

The next subject's topic is created in steps 39 through 41. It uses a topic entry command just as in the previous topic, and text that is inserted from another file. The third subject's topic is created similarly in steps 42 through 44. The text for this topic is entered directly.

The topic for the last subject is created in steps 45 through 48. It is similar to the previous topics, except that it displays a picture with multiple hot spots. The topics that are displayed in popup windows when those hot spots are clicked are then created in steps 49 through 57.

### **Comment**

This How-To also demonstrates inserting text from another file into an application. This is a common and valuable technique, because useful material may be available in another source. Copying and pasting through the Windows Clipboard can work equally well. You can even copy material from Windows Help or Viewer applications this way! Be certain that you only copy material that you have the right to use. All material should be presumed to be copyrighted and restricted unless you created it or the material states that it can be redistributed or published without restriction.

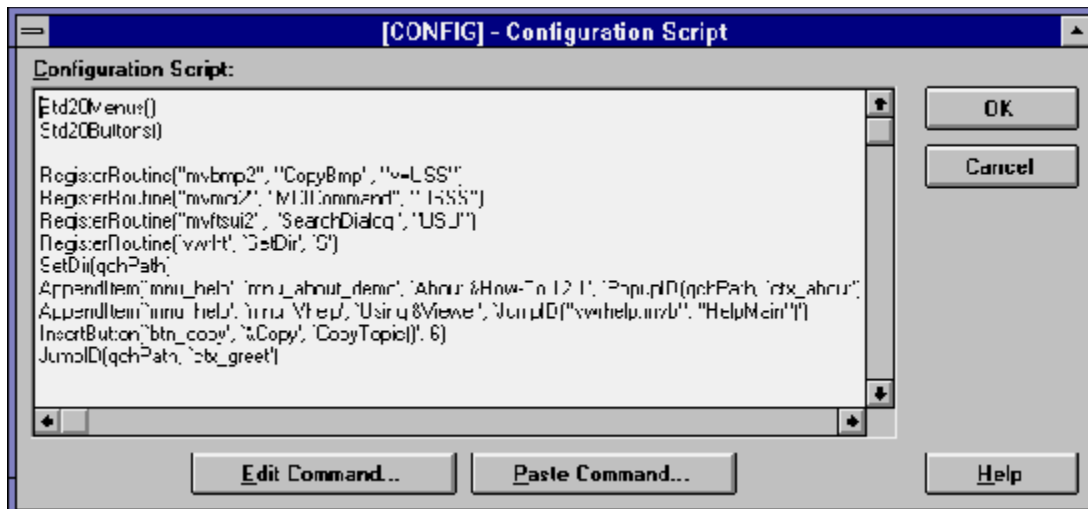
A multiple hot spot picture, such as in the Holding Calls section, might serve as the main menu of a course (or a section of a course). It could show a form, an application screen, a piece of equipment, or any other subject.

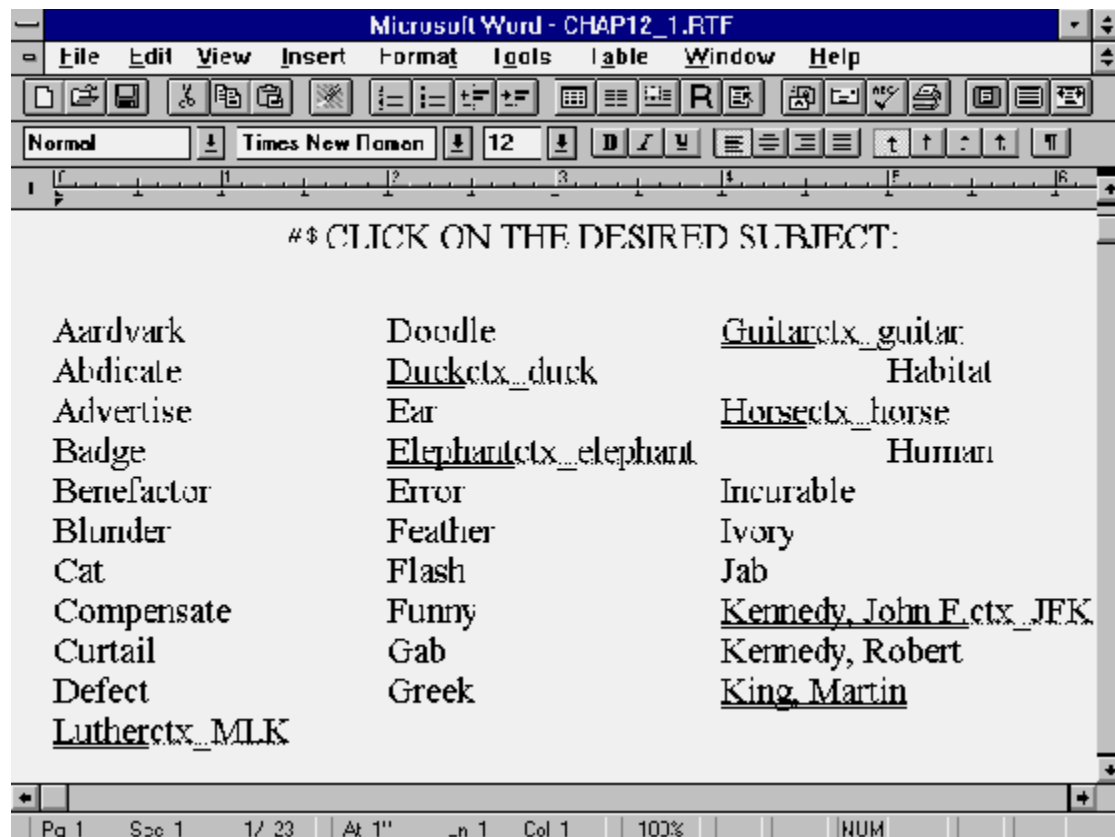
Careful use of movies, animations, and sounds can make a training course more interesting and educational. The examples of proper and improper telephone answering in this demonstration wouldn't be nearly as effective if the user just read them in text!



### 12.3 Tips and Tricks

- ⇒ How-To 12.1 incorporates the sample Viewer help file (VWRHELP.MVB) that is installed with Viewer in the MVHLPUSA subdirectory. To keep the demonstration simple, this file is included exactly as it is distributed. In a real application the file's contents would be customized to reflect only the appropriate techniques. All of the source files used to create this file are included in the same directory.
- ⇒ Be careful not to be annoying with sounds or other effects that can't be avoided. It can be very easy to overdo these effects. Consider the environment where your application might be used. It might be courteous to offer a menu option to suppress sounds, as demonstrated in How-To 6.2.
- ⇒ Large applications should be compiled with maximum compression to reduce space requirements. Compression can also provide faster operation if your application will execute from a CD-ROM.
- ⇒ Viewer's Technical Reference shows how to start multiple instances of Viewer, for operations like the help function in How-To 12.1. The electronic book on the enclosed CD-ROM uses this for help and the Viewer demos.





---

# \$+ (vfid10) **John F. Kennedy** (vfid)

{ewl MVMCI2, ViewerMCI, [device AVIVideo][stdcontrol][caption \pc 'Watch a movie']movies\kennedy.avi}

John F. Kennedy was the 35th President of the United States. He was inaugurated in January 1961 and was assassinated in Dallas, Texas on November 22, 1963. He was the first Roman Catholic to be elected President. The Berlin Wall was built during his administration, in 1961.

---

Microsoft Word - CHAP12\_1.RTF

File Edit View Insert Format Tools Table Window Help

Normal Times New Roman 12

#4+ (vld:0) Saturn(vfld)

{ewl MVBMP2, ViewerBmp2, \nomsg macro="MCICommand(hwndContext, qchPath, 'saturn.wav', 'play')"[caption="\pc; bClick on picture to play music" dither]}saturn.bmp)

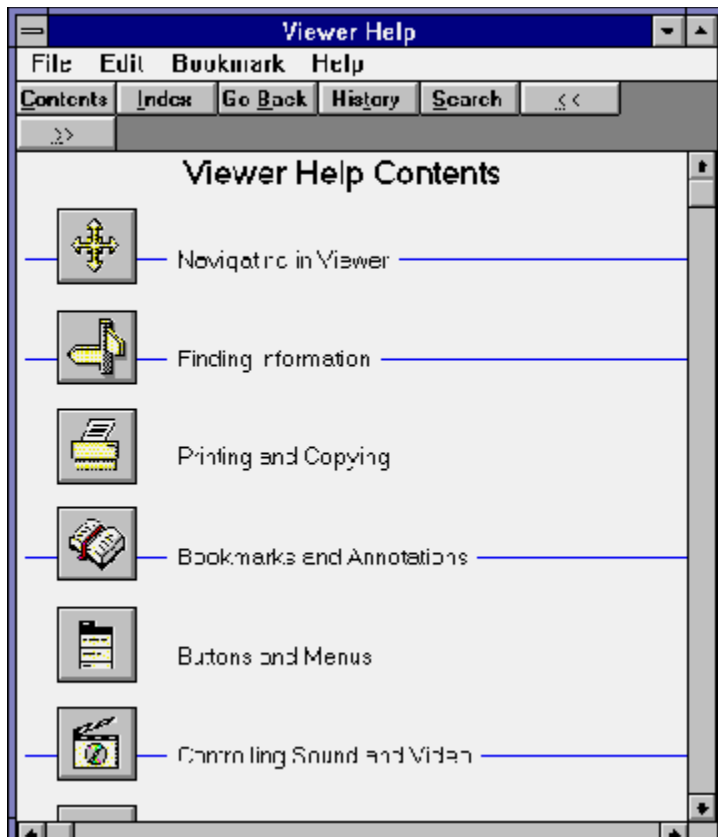
Saturn is the sixth planet from the sun. It has a diameter of 72,000 miles and revolves around the sun once every 29.5 years. It is famous for the series of thin, flat rings that circle it.

#3+ (vld:0) Sun(vfld)

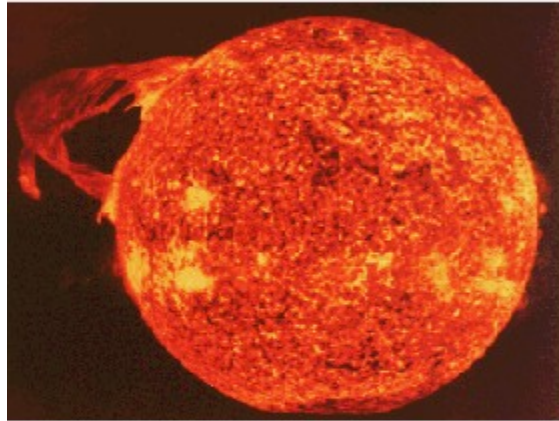
{ewl MVBMP2, ViewerBmp2, \macro="MCICommand(hwndContext, qchPath, 'solar.wav', 'play')"[caption="\pc; bClick on picture to play music" dither]}solar.bmp)

The Sun is a star, just like most of the tiny points of light you see in the sky. The picture shows a phenomenon called solar prominences clearly. The sun is the center of our solar system, providing light and heat to the planets.

Pg 1 Sec 1 1 / 23 At 1" Ln 1 Col 1 100% NUM



## Sun



Click on picture to play music

The Sun is a star, just like most of the tiny points of light you see in the sky. The picture shows a phenomenon called [solar prominences](#) clearly. The sun is the center of our solar system, providing light and heat to the planets.

**Search**

Search by Word

Search In:

All Topic Groups  
 Selected Topic Groups  
 Current Topic Only

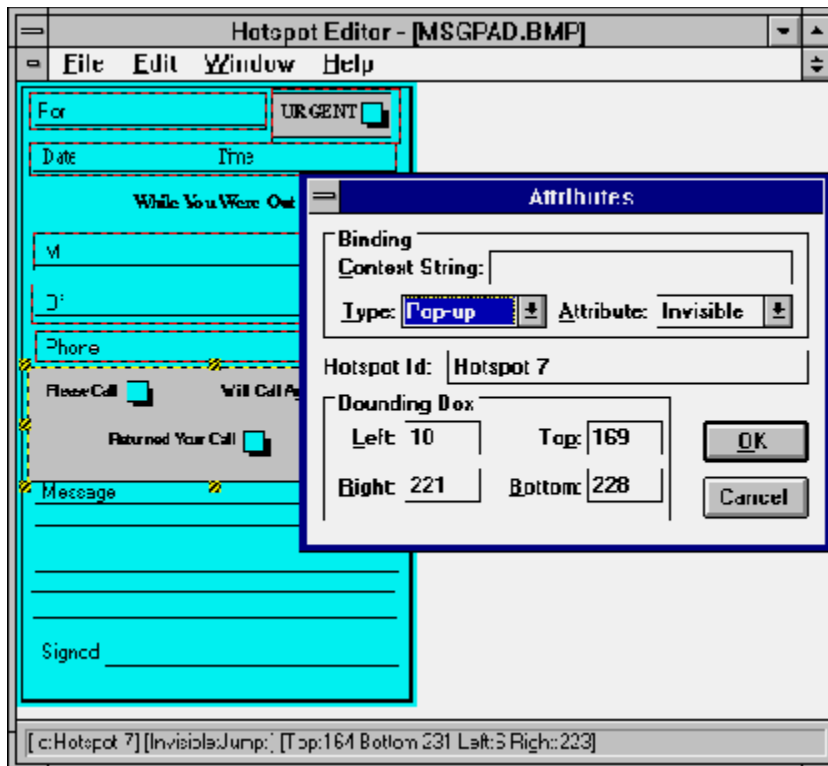
**Topic Groups:**

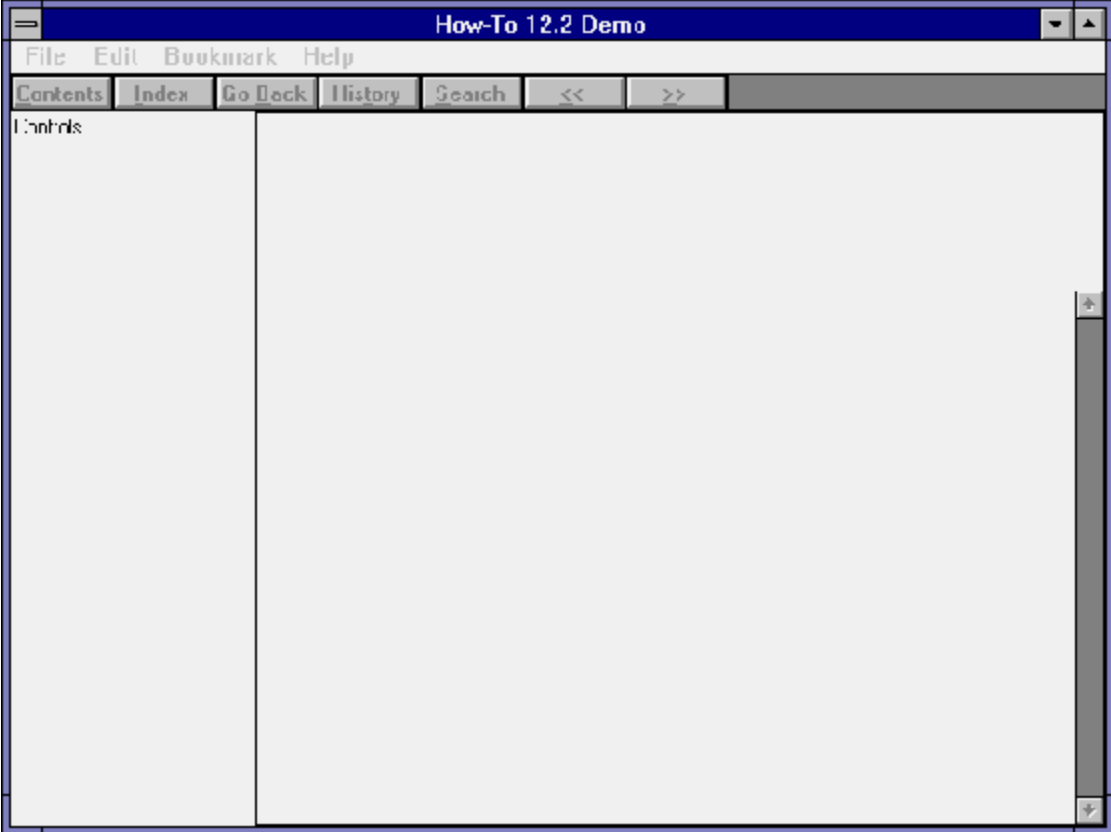
- American History
- Animals
- Space
- Music

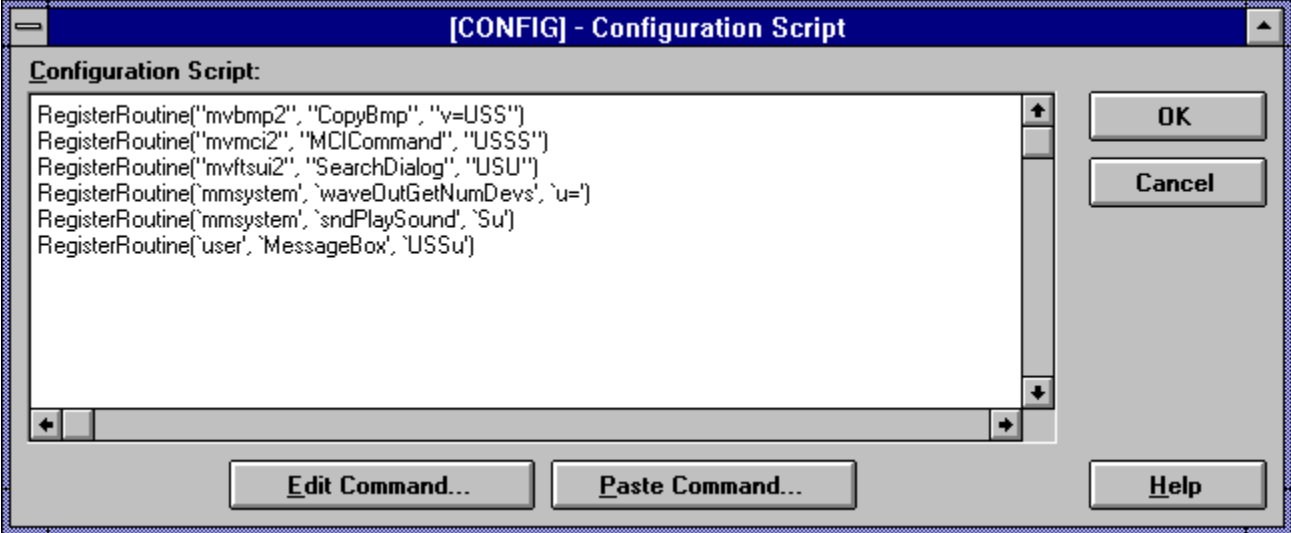
Search by Category

| Category: | Keyword:             |
|-----------|----------------------|
| Subjects  | <input type="text"/> |
|           | Duck                 |
|           | Elephant             |
|           | Guitar               |
|           | Horse                |
|           | John F. Kennedy      |
|           | Martin Luther King   |
|           | Piano                |
|           | Rayline              |









Microsoft Word - CHAP12\_2.RTF

File Edit View Insert Format Tools Table Window Help

Normal Times New Roman 0

## #! Answering the Telephone

What did you think?

Hard to understand? If you were calling,  
how would you feel about a company that  
has someone answering the phone that way?  
Do you want to talk with this person? Would  
you trust a person who gives this impression

{ews  
MVBMP2,  
ViewsBug2,  
!desktop.bmp}

---

**Footnotes** Close

#rx\_greeting  
!IfThenElse(waveOutGetNumDevs(), "sndPlaySound('badans.wav', 1)",  
"MessageBox(hwndApp, 'Your system does not support sound', 'ERROR', 0))

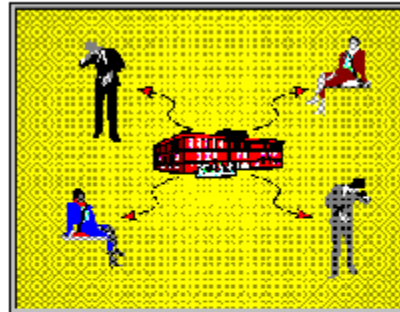
#cx\_instructions  
!PanelID(cchPath, \!a control all>Controls', 0)

#cx\_control\_c1

Pg 1 Sec 1 1/20 At Ln Col 1 100% NUM

```
#{@wc MV3BMP2, ViewerBmp2, [macro='JumpID(qchPath, `ctx_intro`)] !intrc.bmp}
{@wc MV5BMP2, ViewerBmp2, [macro='JumpID(qchPath, `ctx_basics`)] !basicsn.bmp}
{@wc MV5BMP2, ViewerBmp2, [macro='JumpID(qchPath, `ctx_no.c`)] !no.cn.bmp}
{@wc MV5BMP2, ViewerBmp2, [macro='JumpID(qchPath, `ctx_message`)] !msgn.bmp}
{@wc MV5BMP2, ViewerBmp2, [macro='Exit()'] !exitn1.bmp}
```

## You Are the Company!



When a customer calls for the first time, your answer forms their first impression of the company. You are the company! It's about how you react to the way different companies answer the phone.

Try to be..

**COURTEOUS** - be polite, and never argue

**CHEERFUL** - a friendly voice is always pleasant

**CONFIDENT** - convince callers you'll handle the request properly

**CLEAR** - be sure the caller understands you

Index

Basics

Hold

Messages

Exit



Anyone who has used Windows for a while is probably familiar with the normal procedure for installing software. You insert the first disk in its drive, and use the Program Manager to run A:\SETUP (or B:\SETUP). You respond to a few dialog boxes confirming the names of directories to be used and components to be installed, and then the files are copied. You are prompted to insert additional disks as required.

The Setup program used by most of these applications was originally developed by Microsoft to install its own applications. The Setup program is now available in the form of a toolkit in the Software Development Kit (SDK) for the use of developers. A limited version is distributed with Viewer. This is designed—like Viewer itself—to be used without extensive technical knowledge. If you need to perform functions that aren't included in the Viewer version, you should purchase the SDK. Other installation packages are also available through mail-order companies and retail stores that cater to developers. Some programming skills may be needed to use these more advanced packages.

This appendix provides an overview of the {vfld137438953482}Setup{vfld-9223356093936173056} toolkit included with Viewer, and the additional features available in the full toolkit. This is not complete documentation for either version. The Viewer *Authoring Guide* and the SDK's *Setup Toolkit for Windows* provide more information. You can get more assistance with either version of Setup through CompuServe, as described in appendix C.



## How Does Setup Work?

Setup can perform a number of common functions, including:

- ü Copy application files to new or existing directories
- ü Determine if there is enough disk space available to hold the selected files
- ü Check file dates and version numbers to avoid copying older versions of files over newer ones
- ü Copy system files to the Windows and Windows System directories
- ü Install new fonts or drivers
- ü Create a new Program Manager group and items
- ü Update the WIN.INI and SYSTEM.INI files
- ü Create or update private INI files
- ü Exit and restart Windows if necessary to activate new system files, fonts, or drivers

The Setup operation is controlled by a program that must be written using Setup's language, which is very similar to Basic. This can be as simple as a script or as complicated as an advanced program. Most of the installation functions are controlled through reasonably simple programming, using standard program commands that are provided with Setup. Some operations require complex programming within Setup or in external commands in DLLs. An installation script is provided in the version of Setup included with Viewer.

Two files control most of Setup's operations. The INF file contains information about the files being installed and the disks containing those files, and the MST file contains the Setup script. The versions of these files included with Viewer are named TITLE.INF and TITLE.MST. The comparable files created by the Setup included with the SDK are named SETUP.INF and SETUP.MST.

The MST file script includes commands that read the information contained in the INF file, and other commands that perform the desired functions. The commands can execute Windows API commands, execute external commands written specially for the installation operation, display dialog boxes that let the user select desired installation options, and perform other functions needed to produce the desired results.

Prepare Setup for your application by copying the standard INF and MST files into a directory, and customizing them as appropriate for your application. Then create a directory structure that represents the distribution disks, and copy the Setup and application files into the proper directories. You can test your design by executing Setup from these directories, or by copying the files to the individual disks and running Setup from those disks.

Setup creates a temporary directory on the drive that contains the active Windows directory, and copies the files needed to perform the installation into that directory. The MST program is then executed, which usually verifies the available disk space, determines the target directory name, and copies the files to be installed. A dialog box shows the progress as the files are copied, and various bitmaps can be displayed in the dialog box as the installation proceeds.

The INF file is arranged much like an MVP or INI file—it has section headings enclosed in brackets followed by detail entries. The TITLE.INF file has four sections: Source Media Descriptions describes the distribution disk, Default File Settings sets the default values for installation options, System Files describes Viewer’s runtime files, and Installed Title Files describes files created by the author. Entries in the sections that describe files consist of one line per file, containing a series of fields separated by commas. The standard definition in TITLE.INF is as follows:

```
disc_number,filename,,,,file_date,,,,,,,,,file_size,,,file_version,
```

All commas must be included, even for unused fields. Viewer’s *Authoring Guide* describes only a few of the fields that can be used. The full description is shown in Table A-1. The fields with names that have STF\_ prefixes contain default values that are set in the INF file.

**Table A–1. INF File Definition Fields.**

| Item            | Value       | Description   |
|-----------------|-------------|---|
| 1 - Disc_Number | integer     | Number of the disk containing this file. Use 1 for all files on a CD-ROM. |
| 2 - Filename    |             | Name of the file to be copied, including relative path from SETUP.EXE.    |
| 3 - Append      | <empty>     | Overwrite any existing file.  |
|                 | filename    | Append to specified file. Can’t use with Rename, Root, or Backup.         |
| 4 - Backup      | <empty>     | Use default value in STF_BACKUP.  |
|                 | *           | Rename old file with BAK extension.                                       |
|                 | filename    | Rename old file to specified name.  |
| 5 - Copy        | <empty>     | Use default value in STF_COPY.  |
|                 | COPY        | Copy this file.   |
|                 | !COPY       | Don’t copy this file.   |
| 6 - Date        | <empty>     | Use default.  |
|                 | YYYY–MM–DDD | Set file date to specified date.  |
| 7 - Decompress  | <empty>     | Use default value in STF_DECOMPRESS.                                      |
|                 | DECOMPRESS  | Decompress the file.  |
|                 | !DECOMPRESS | Don’t decompress the file.  |
| 8 - Destination | <empty>     | Unsupported—must be empty.  |
| 9 - Overwrite   | <empty>     | Use default value in STF_OVERWRITE.                                       |
|                 | ALWAYS      | Always replace existing file with the same name.                          |
|                 | NEVER       | Never replace an existing file.   |
|                 | OLDER       | Only replace existing file with lower version number or older date.       |
| 10 - ReadOnly   | UNPROTECTED | Only replace file with Write attribute.                                   |
|                 | <empty>     | Use default value in STF_READONLY.  |
|                 | READONLY    | Set read-only file attribute.   |
| 11 - Remove     | !READONLY   | Don’t set attribute.  |
|                 | <empty>     | Copy the file using other properties (do not remove).                     |

|                   |                                |  |
|-------------------|--------------------------------|--|
|                   | REMOVE                         | Remove the file from hard drive (do not copy).   |
|                   | !REMOVE                        | Copy the file using other properties (do not remove).  |
| 12 - Rename       | <empty><br>filename            | Leave file name unchanged.<br>Rename file to specified name. This can include a subdirectory name.                             |
| 13 - Root         | <empty><br>ROOT<br>!ROOT       | Use default value in STF_ROOT.<br>Strip subdirectories from file name.<br>Leave subdirectories in file name.                   |
| 14 - SetTimeStamp | <empty><br>SETTIME<br>!SETTIME | Use default value in STF_SETTIME.<br>Use the Date property.<br>Use the current system time.                                    |
| 15 - Shared       | <empty><br>SHARED<br>!SHARED   | Do not treat file as shared.<br>An existing version of file may be in use during installation.<br>Do not treat file as shared. |
| 16 - Size         | integer                        | Uncompressed size of file in bytes.  |
| 17 - System       | <empty><br>SYSTEM<br>!SYSTEM   | This is not a system file.<br>System file; will be replaced during system restart.<br>This is not a system file.               |
| 18 - TimeToCopy   | <empty><br>integer             | Use default value in STF_TIME.<br>Progress indicator increment.  |
| 19 - Reserved     | <empty>                        | Leave this field empty.  |
| 20 - Version      | <empty><br>Version             | No version number.<br>One to four integers separated by periods.   |
| 21 - Vital        | <empty><br>VITAL<br>!VITAL     | Use default value in STF_VITAL.<br>Installation fails if file cannot be installed.<br>Installation can continue.               |

---

## Setup in Viewer

This version of Setup uses a simplified script. It is designed specifically to install Viewer applications from a CD-ROM without requiring extensive technical or programming knowledge. The MST file includes comments that show you the changes needed to

- Install multiple MVB files at once
- Install Help MVB files
- Install custom DLL files
- Install custom fonts
- Create multiple Program Manager items
- Display custom icons for Program Manager items
- Install Video for Windows runtime files

The script includes instructions for other options as well—you should read it. It does not show you how to replace the Viewer Title Setup banner (this requires C programming), or to use diskettes or compression (this requires the full Setup toolkit from the SDK).

A complete sample Viewer installation can be found on the enclosed CD-ROM disk in the MVSAMPLE\GALLERY\IMAGE directory. This includes all files needed to install the Gallery sample application, including updated INF and MST files.

English, French, and German versions of the Setup files are included (in the USA, FRN, and GER subdirectories within MVSAMPLE\PROGSAMP\SETUP), as well as corresponding versions of the Viewer runtime files (in the MVSYSUSA, MVSYSFRN, and MVSYSGER subdirectories). Note that some error messages are hard-coded into the Setup programs, and are only displayed in English.

The MST script includes commands to update or create the VIEWER.INI file in the user's Windows directory. The Files section of the INI file includes a line for each MVB or DLL file. These entries are used by Viewer to locate files if they are not found in the current, Windows, or Windows System directories, or directories in the Path. These entries also include a one-line message (of less than 50 characters) that is displayed if the file cannot be located. This is especially useful for MVB files that are read from CD-ROM disks. The format of these entries is simply the MVB filename, an equals sign, the path, a comma, and the error message. An example of such an entry is the following:

```
VIEWERHT.MVB=H:\VIEWERHT\,Please insert the Viewer How-To disk
```

The Viewer Setup expects your MVB file to remain on the CD-ROM. If you plan to copy it to the hard drive, the code that creates the VIEWER.INI entry must be changed. To do this, follow these steps:

1. Locate the line that reads GLOBAL CUIDLL\$. Insert a new line immediately after that reads GLOBAL szTitleDir\$.
2. Locate the SUB ModifyViewerIni STATIC entry.
3. Locate a comment that starts *Now we have to register the MVB file.*
4. In the command following that comment, replace the portion that reads *GetSymbolValue("STF\_SRCDIR")* with szTitleDir\$. This inserts the

target drive and directory in the INI entry instead of the source drive and directory.

5. Include an entry for the MVB file in the Installed Title Files section of the TITLE.INF file. This causes the file to be copied.

## Setup in the SDK

The full version of Setup includes two programs that help create the INF file and the directories that represent the distribution disks. These programs compress files, split files that are too large to fit on a single disk, and lay out which files should be on which distribution disks. They let you control all of the options recorded in the INF file through standard dialog box features such as check boxes and radio buttons. The full Setup also provides the complete set of functions that can be executed in the MST script.

Two different compression systems are available—external and internal. Most applications use the standard external compression, which uses an external program (LZEXPAND.DLL) to decompress files. This lets users decompress and install individual files manually if necessary. However, external compression is very inefficient for large files that must be split between disks. The file is first divided into parts the size of a distribution disk, and then compressed. The space freed up by compression is not automatically used for other files, which means it is usually wasted. This makes the program call for unnecessary extra distribution diskettes.

Files compressed with internal compression can only be decompressed by the Setup program. Files that use internal compression are laid out much more efficiently on the diskettes, with far less wasted space. Internal compression cannot be used for SETUP.EXE, SETUP.LST, or for driver files that must be installed through the Control Panel.

## Tips and Tricks

- ⇒ The Setup Toolkit is very powerful, but poorly documented. As a result, problems are common. If you cannot avoid these problems by using the prepared script supplied with Viewer, you can get assistance through the WINSDK forum in [{vfld137438953482}CompuServe{vfld4926370438184960}](#). This is described in Appendix C.
- ⇒ Setup includes a function, [{vfld137438953484}GetCopyListCost{vfld2378181537062453248}](#), that determines the disk space required by the installation. It allows for disk cluster sizes and the difference in sizes of files being replaced, and excludes files that will not be copied based on the Overwrite option. However, it does not handle split files properly. More details are available through CompuServe by entering G MSKB to select the Microsoft Knowledge Base, then following the prompts to select document Q88749. Download this document to read offline.
- ⇒ The GetFreeSpaceForDrive and GetCopyListCost functions return sizes measured in bytes. Given the gigabyte drives now found on some PCs, be prepared for *very* large numbers!
- ⇒ If your files are installed on a system using disk [{vfld137438953482}compression{vfld280933810831360}](#) (such as Stacker or SuperStor), the free space reported assumes that your files can be compressed to the same degree (about 50 percent) as all other files. If you are installing large files that cannot be compressed effectively (such as AVI movies), this could report that there is sufficient space even though there isn't really enough. The best solution, if you have such files, is to warn your users of this possibility.
- ⇒ To change the Microsoft Setup logo, first create a monochrome bitmap using white for the text or image and black where the blue background should show through. Then change DIALOGS.RC in the BLDCUI directory to use your file instead of BITMAP.DIB and recompile MSCUISTF.DLL with the modified resource file. Be sure to use copies of these files in a different directory to avoid changing the standard files.
- ⇒ The blue background cannot be changed—it is hard-coded into the program.
- ⇒ To recompile MSCUISTF.DLL using Microsoft Visual C++, ignore the makefile in the BLDCUI directory. Instead, create a new project in VC++. Choose Project from the Options menu, then select Windows DLL as the project type, and choose Release as the build mode. Both of these project options are critical. Accept all defaults for compiler options. Add the following files to the project: DLGPROCS.C, MSCUISTF.DEF, MSCOMSTF.LIB, MSSHLSTF.LIB, MSUILSTF.LIB, and DIALGS.RC. You can then recompile successfully.
- ⇒ If you have a hard-to-find error in your Setup script, one way to pin it down is by inserting pause statements at strategic points. The syntax is PAUSE string\$, so you could use a call like PAUSE

"Directory " + szTitleDir\$. The string is displayed in a message box with an OK button.

- ⇒ The date and time stamps of an installed file are determined as follows:
  - If the SETTIME option is applied, the file date is taken from the file on the distribution disk, and the time is set to 12:00 a.m.
  - If the !SETTIME and DECOMPRESS options are applied and external compression is specified, then the file date and time from the distribution disk are used. This is the only way to control the file's time stamp. If internal compression is used, the file reflects the time it is copied.
  - If the !SETTIME and !DECOMPRESS options are applied, the file shows the date and time it is copied from the distribution disk. Note that you can flag uncompressed files to be decompressed. It doesn't hurt.
- ⇒ If your script executes CreateProgmanItem with cmo% set to cmoOverwrite and szOther\$ containing any value, that command creates a new item instead of replacing the existing one. This is a known bug in Setup. This means you can't use cmoOverwrite while specifying an icon, startup directory, etcetera. Assistance can be obtained through the WINSKD forum on CompuServe.
- ⇒ The Setup documentation states that the /W option for the Dsklayt2 program (part of the full SDK toolkit) can be used without a disk number. This is incorrect—a disk number is required.
- ⇒ If you want to maximize the Setup window, add the following instructions to your MST file:

```
CONST WS_VISIBLE = &H10000000
CONST WS_BORDER = &H00800000
CONST WS_CLIPCHILDREN = &H02000000
CONST GWL_STYLE = -16
CONST SW_SHOWMAXIMIZED = 3
...
DECLARE FUNCTION
{vfld137438953483}ShowWindow{vfld3131967461654528} LIB
"user.exe" (hWnd%, iShow%) AS INTEGER
DECLARE FUNCTION {vfld137438953483}SetWindowLong{vfld-
9042521606949175296} LIB "user.exe" (hWnd%, offset%, style&) AS
LONG
...
hWnd% = HwndFrame()
i& = SetWindowLong(hWnd%, GWL_STYLE,
WS_VISIBLE+WS_BORDER+WS_CLIPCHILDREN)
j% = ShowWindow(hWnd%, SW_SHOWMAXIMIZED)
```

If the calls are made at the beginning of the script there should be only a quick flicker as the window size and border are changed.

- ⇒ The Dsklayt2 program (part of the full Setup toolkit) supports an option, /V, that is missing from the documentation. This causes the program to produce detailed progress messages that can be invaluable for debugging.
- ⇒ If you want to place your files in directories in the distribution diskettes, change the file name field entries in the INF file to the form DIRECTORY\FILENAME, for example, SYSTEM\TSTOOLS.DLL. This must be done manually—the Dsklayt program does not support this. You may be able to make these



changes by using Word's search and replace function. Use the ROOT option if the directories should not be created on the target drive.

- ⇒ The recommended way to create files in multiple directories is to put the files for each directory in a separate section in the INF file. Then, in the MST file, use AddSectionFilesToCopyList for each section and set the szDest\$ parameter to the desired directory name. Use CopyFilesInCopyList to create the directory and copy the files.
- ⇒ Be sure to read the README.WRI, TESTDRV.HLP, and FILEDESC.WRI files included with the full Setup toolkit. They contain information that is extremely important.



The CD-ROM enclosed with this book is organized into more than 300 directories containing more than 1700 files. The files contain nearly 150 megabytes of information—enough to fill almost 100 3.5-inch diskettes! This appendix describes the contents of each directory on the disk, and also describes Viewer extensions, provided by other developers, included on the disk. Instructions for installing the files on your computer are found in the Quick Start section at the front of this book. The files fall into four major categories:

- ü A special version of the Microsoft Viewer Publishing Toolkit
- ü The source and compiled files that are used or created by each How-To in this book
- ü The complete text of this book, as a Viewer application
- ü Software that extends the capabilities of Viewer

This great mass of material is carefully organized to help you find any files you need. The directory structure is shown in Table B-1. There are two major groups of directories—the Viewer Toolkit itself, and the material related to this book. The Toolkit occupies the root directory and the first 14 top-level directories listed. The material related to this book is grouped in suitable subdirectories under the VIEWERHT directory. To avoid unnecessary duplication, the subdirectories for the individual How-To sections are represented by a single typical entry. In particular, the directories for chapters 3 through 12 are represented by the single CHAP $x$  entry, and the almost 70 individual sections are represented by the single CHAP $x_y$  entry. Each chapter's directory contains the subdirectories for that chapter's sections. Each section's subdirectory contains the project files and four standard subdirectories, as shown in Table B-1.

**Table B-1 Directory Structure of Viewer How-To CD**

|            |   |
|------------|---|
| \          | Root directory—Viewer installation files    |
| 1 GRPHFLT  | Graphic import filter files                 |
| 1 MVHLPFRN | Viewer Help MVB and project files (French ) |
| 1 GRAPHICS | Graphics source files                       |
| 1 TEXT     | Text source files                           |
| 1 MVHLPGER | Viewer Help MVB and project files (German ) |
| 1 GRAPHICS | Graphics source files                       |
| 1 TEXT     | Text source files                           |
| 1 MVHLPUSA | Viewer Help MVB and project files (English) |
| 1 GRAPHICS | Graphics source files                       |
| 1 TEXT     | Text source files                           |
| 1 MVSAMPLE | Sample files:                               |
| 1 GALLERY  | Gallery MVB and project files               |
| 1 GRAPHICS | Graphics source files                       |
| 1 IMAGE    | Sample Setup directory                      |

|            |            |            |   |
|------------|------------|------------|---|
|            |            | 1 SOUNDS   | Sound source files                          |
|            |            | 1 TEXT     | Text source files                           |
|            |            | 1 VIDEO    | Video source files                          |
|            | 1 PROGSAMP |            | Programming samples                         |
|            |            | 1 EPLIST   | Embedded window DLL                         |
|            |            | 1 KATASRCH | Custom search function                      |
|            |            | 1 SETUP    | Setup directories                           |
|            |            | 1 FRN      | Setup files (French)                        |
| BLDCUI     |            |            | 1   |
|            |            | 1 GER      | (French) Setup files (German)               |
| BLDCUI     |            |            | 1   |
|            |            | 1 USA      | (German) Setup files (English)              |
| BLDCUI     |            |            | 1   |
|            | 1 USA      |            | (English) Setup DLL source                  |
|            |            | 1 MEDIA    | USA Tour MVB and project files              |
|            |            | 1 PCPICS   | Sound and movie files                       |
|            |            | 1 SONYHELP | Picture files (Windows)                     |
|            |            | 1 HELPPICS | Help text file (Sony)                       |
|            |            | 1 SONYPICS | Help picture files (Sony)                   |
|            |            | 1 TEXT     | Picture files (Sony)                        |
| 1 MVSONY   |            |            | Text files                                  |
| 1 MVSYSFRN |            |            | Sony authoring files                        |
| 1 MVSYSGER |            |            | Viewer runtime files (French)               |
| 1 MVSYSUSA |            |            | Viewer runtime files (German)               |
| 1 MVTOOLS  |            |            | Viewer runtime files (English)              |
| 1 SYSTEM   |            |            | Viewer authoring and support files          |
|            |            |            | Files installed in Windows\System directory |
| 1 TUTORIAL |            |            | Files for Viewer tutorial                   |
| 1 WINDOWS  |            |            | Files installed in Windows directory        |
| 1 XA       |            |            | Files for Sony application                  |
| 1 VIEWERHT |            |            | Material related to this book               |
|            | 1 ADDONS   |            | Viewer extensions:                          |
|            |            | 1 MVVBX    | Viewer Visual Basic Custom Control (VBX)    |
|            |            | 1 PUSH     | Viewer Button command                       |
|            |            | 1 SCORE    | Question Scoring command                    |
|            |            | 1 VBCOMM2  | Viewer Commander                            |
|            |            | 1 VLAUNCH  | Viewer 1.0/2.0 Launcher utility             |
|            |            | 1 TSTOOLS  | TSTools DLL                                 |
|            | 1 HOWTOS   |            | How-To section files                        |
|            |            | 1 CHAPx    | Directories for chapter x (CHAP3-CHAP12)    |
|            |            | 1 CHAPx_y  | Project files for chapter x section y       |
| MOVIES     |            |            | 1   |
|            |            | 1          | Movie and animation files                   |
| PICTURES   |            |            | 1   |
|            |            | 1          | Picture (BMP and SHG) files                 |

|        |          |  |        |   |
|--------|----------|--|--------|---|
|        |          |  | 1      | Wave and MIDI files                         |
| SOUNDS |          |  | 1 TEXT | Text (RTF) files                            |
|        | 1 SYSTEM |  |        | Files installed in Windows\System directory |

This CD contains a special version of the Microsoft Viewer Publishing Toolkit, with a limited compiler that was prepared for this book. This version of the compiler restricts projects to no more than 25 topics, and adds the words *Demo Version* to the title bar of the compiled applications. All supporting programs, demonstrations, help files, and other material in the commercial toolkit are included unaltered. The root directory also contains the setup program and files used to install Viewer.

The VIEWERHT directory is divided into three subdirectories. The ADDONS subdirectory contains the extensions to Viewer—each in its own subdirectory. The HOWTOS subdirectory contains one directory for each chapter in the book—each contains one subdirectory for each How-To section in that chapter. Each of these How-To subdirectories contains all of the files used or created by the corresponding section, including the compiled result. The subdirectory for each How-To uses the standard structure, with subdirectories for movies, pictures, sounds, and text files. All of these subdirectories are created, even if some of them are not used. The SYSTEM subdirectory contains the files that are installed in the Windows\System directory. This includes Windows MCI drivers required by How-To demonstrations and copies of the Viewer extension executable programs. The VIEWERHT directory also contains the contents of this book in the form of a Viewer application (VIEWERHT.MVB), and the setup program and files used to install these applications.

### Viewer Extensions (Add-Ons)

Each extension is located in its own subdirectory, containing the executable program, documentation files, and a demonstration if appropriate. Refer to these files for more information. The executable files for these extensions are installed in your Windows\System directory as part of installing the files for this book. The other files may be copied onto your hard drive from the CD-ROM if desired.

These extensions were created by Viewer developers, who made them available through this book in the hopes of fostering development with Multimedia Viewer. Unless stated otherwise, you must obtain a license to use any of these extensions in a commercial application or if you want support. They may be used for development or for in-house applications without a license. Please contact the appropriate developer, at the address shown in appendix D, for licensing information. These developers also have other functions available.

|                       |   |
|-----------------------|---|
| Viewer Custom Control | Lets Visual Basic and Microsoft Visual C++ programs respond to Viewer events such as jumps or scrolling, obtain information about Viewer apps that are running, |
|-----------------------|---|

and read Baggage files. This also simplifies issuing commands to a Viewer application from a program. Includes MVSPY demo program, which can monitor the operation of any Viewer application. The demo requires the CMDIALOG.VBX to compile. You can examine the code by ignoring error messages while loading the project into VB or VC++. Developed and copyrighted by Keyboard Publishing.

Button command

Lets you define a special hot spot within any topic. It uses two bitmaps to represent normal and depressed button images, or two alternate toggled states (like a light switch). Note: this is an unsupported and unfinished product—use it at your own risk. A commercial version is being developed. Developed and copyrighted by Keyboard Publishing.

Question scoring command

Lets you calculate and display the user's score (percentage of correct answers) on multiple choice questions. Developed and copyrighted by Keyboard Publishing.

Viewer Commander

Lets you execute Viewer commands (within a running application) from a command line or from a script file. This lets you modify parts of your application without recompiling. Developed and copyrighted by Keyboard Publishing.

Viewer Launcher utility

Executes the correct version of Viewer—1.0 or 2.0—for an application. This may be valuable if users are still using Viewer 1.0 applications. A license is not required, and a demonstration is not included. Developed and copyrighted by Keyboard Publishing.

## TSTools DLL

Provides commands that let you do the following:

- (a) Create a button in an embedded pane within any topic, which executes Viewer commands when clicked. You can specify the button size, and a caption in selected font, size, and style.
- (b) Call WinHelp for a specified file and context string.
- (c) Create a custom dialog box with up to three lines of text. This can serve as an About box, for example.
- (d) Execute Windows Write and copy the current topic into Write. Additional calls copy other topics into the same instance of Write.

The following functions in this DLL are included and demonstrated, but are not documented here or licensed for use:

- (e) Execute a command after an MCI command completes, such as jumping to a new topic after a sound or video completes.
- (f) Execute a command after a specified delay. A series of delays and associated commands can be defined. This can be used to create a self-running demo.

Complete documentation and additional functions are available by licensing these extensions from the developer. Developed and copyrighted by TouchSend Management Consulting.





Since Viewer is a very sophisticated and powerful system, you might want to ask for some help. You might have questions about technical issues, design issues, or any other aspect. You might be stuck, and want advice from an experienced Viewer author. You might even find a bug in one of the programs, and want to report it or obtain a corrected version. You'll be happy to know that every bit of this is readily available, through a service known as CompuServe.

## What Is CompuServe?

CompuServe is an online service operated by a private company. It provides facilities for users to communicate with each other through public messages that all users can read, and private messages that can only be read by the addressee. It also allows files to be loaded into public libraries, and for those files to be copied to your computer upon request.

CompuServe is a huge operation. It has many users throughout the world, with widely diverse interests. It is also used by many companies, including Microsoft, as a means of providing support to their customers. Obviously, you don't want to sift through thousands of messages looking for the ones that interest you. CompuServe keeps the system usable by dividing it into many separate areas known as *forums*, which are further divided into *sections*. Each forum and section is dedicated to a particular area of interest, so you can look at just the messages and files in the sections that interest you. Microsoft operates many forums to provide assistance to its customers for all of its programs.

To use CompuServe, you need a modem and a communications program. The modem is connected to your telephone line, which permits it to call the nearest CompuServe location. The program controls the modem and all of the technical communications details. You also need to join CompuServe and get your own membership number and password.

CompuServe charges you a fee for every minute that you are connected to its service. You also have to pay for the phone call, but this will probably be a local call if you live in a city. Several computer programs have been developed that help you minimize these costs by reducing the time you need to be connected to CompuServe to do what you want. These programs will read messages you select at high speed, and record them on your computer for you to read when you are no longer connected to the service. You can then write replies or new messages at your leisure, and the program will transmit them at high speed when you are ready. You can buy a special CompuServe kit, containing one of these programs and complete instructions, at most computer software stores and bookstores. These kits cost about \$25, and provide a \$25 credit toward your CompuServe bill. Other access programs are available as shareware from CompuServe's libraries. These are programs you can try out before paying the license fees.

CompuServe charges \$8.95 per month for basic services, plus connect-time rates based on the speed of your modem connection: speeds up to 2400 bps cost 8 cents per minute and speeds up to 14,400 bps cost 16 cents per minute. Some special services have additional charges.

If you are preparing to develop applications using Viewer you should not hesitate to join CompuServe. It provides an invaluable source of information and assistance.

## **The Viewer Section**

Among the many CompuServe forums and sections operated by Microsoft is a section dedicated to Viewer—Section 6 of the Windows Multimedia (WinMM) forum. Microsoft representatives knowledgeable about Viewer are available here to provide assistance. If necessary, they can refer questions to the programmers who developed and maintain Viewer. Many experienced Viewer authors also monitor the messages here, and will gladly help you. This is also where Microsoft provides corrected versions of Viewer programs.

Microsoft and some independent developers have loaded a number of files into the library for this section. These include external functions that enhance the abilities of Viewer, sample programs for performing functions that have been of interest to many people, and other contributions. You can copy any of these to your computer for just the cost of the connect time. Some of these programs might require paying a license fee if you want to use them in a commercial application, or might be shareware programs that you can try before paying a license fee. Any license requirements are normally clear in the file description that you see before deciding to download the files.

## **Other CompuServe Forums and Sections**

Other forums and sections that could be of particular interest to Viewer authors include Microsoft support for Word for Windows and other Windows applications, and forums operated by several magazines. The Setup program, described in appendix A, is supported in Section 16 of the WinSDK forum. There are also forums or sections operated by the major PC vendors. If you have a problem with your computer, video board, printer, or any other related equipment you may be able to get help here. There are literally hundreds of other forums covering technical areas such as telecommunications, MIDI music, and CD-ROM, and other interests ranging from travel and motorcycles to cancer and human sexuality. There are groups for nearly every interest, and more are formed every month.

## **The Microsoft Knowledge Base**

Microsoft also makes available, through CompuServe, access to its database of known problems, workarounds, technical articles, and similar information for all of their products. This is the database used to answer most telephone calls from its customers. You can examine this database by entering G MSKB at a CompuServe prompt, then following the menu directions.

## **Other Sources of Assistance**

In addition to the CompuServe service, Microsoft also markets a quarterly subscription to its Developer Network CDs. These disks include the complete Knowledge Base and over 100,000 pages of documentation, magazine articles, books, sample programs, and other useful material. This service is designed primarily to support software developers, and it is primarily dedicated to programming issues. It includes a large volume of information

for people writing Windows Help applications, and similar material for Viewer is likely to be included in the future. This subscription may be ordered by calling Microsoft at (800) 759-5474 or faxing (303) 443-5080.



Now that your appetite for Viewer has been whetted, you should be all set to go buy some software, hire some Viewer programmers, or maybe hire a Viewer author. At least, you would if you knew where to find them.

This appendix will help you get started.

## **Software Packages**

Microsoft Multimedia Viewer Publishing Toolkit (Viewer)  
Microsoft Windows 3.1 (Windows)  
Microsoft Windows Software Development Kit (SDK)  
Microsoft Word for Windows (Word)

NOTE: Many of these packages are also available in various upgrade and combination offers. The SDK is also included with some compiler packages. Word and Windows are readily available through local software stores, mail order distributors, and other sources.

### **Microsoft Corp**

Microsoft developed these packages. They sell only for full retail price.

Contact:

(800) 227-4679 (or local Microsoft office outside the United States)

### **Programmer's Connection**

A mail-order retailer that emphasizes supporting developers.

Contact:

7249 Whipple Avenue N.W.

North Canton, OH 44720-7143

Telephone: (800) 336-1166 or (216) 494-8715

Fax: (216) 494-5260

## **Viewer Professional Services**

The companies and individuals listed below were active members of the beta test team for Viewer 2.0. They can assist with consulting, contract programming, training, and authoring services. These and other sources of assistance can be reached through the Viewer section of CompuServe's WinMM forum.

### **Computer Office Procedures**

Developer of a DLL that indexes French verbs by their root form, regardless of the sex and tense of the verb in the text, and author of Viewer applications in both French and English. The only active member of the Viewer beta test team in France.

Contact:

Robert Hester

1 rue des Mouttes  
31750 Escalquens, FRANCE  
Telephone: (33) 61 81 19 27  
Fax: (33) 61 52 05 44  
CompuServe: 100010,2361

### **Keyboard Publishing**

An electronic book publishing company specializing in multimedia in Windows and Macintosh systems.

Contact:

482 Norristown Road, Suite 111  
Blue Bell, PA 19422  
Telephone: (800) 945-4551 or (215) 832-0945  
Fax: (215) 832-0948  
CompuServe: 71151,253

### **Stephen Pruitt**

Author of this book, author of Viewer and WinHelp applications, and consultant assisting in Viewer and WinHelp authoring.

Contact:

through Waite Group Press, or  
CompuServe: 70244,365

### **TouchSend Management Consulting Inc.**

A company that specializes in custom programming to create Viewer tools, designing systems to convert data that needs frequent updating (such as catalogs) into Viewer format, and consulting and training in Viewer authoring. Jeff is internationally recognized as one of the leading experts in the design and implementation of extensions for Windows Multimedia Viewer and WinHelp.

Contact:

Jeff Kovitz  
1904 Chatsworth Way  
Tallahassee, FL 32308  
Telephone: (904) 668-6180  
CompuServe: 76064,3410

### **Clay VerValen**

Former Product Manager for Multimedia Viewer and Interactive TV. Clay is available to consult on all phases of multimedia title development.

Contact:

CompuServe: 72070,2056

## **Other Sources**

### **Green Vine MultiMedia Studios**

Green Vine has a vast library of original music in both MIDI and Wave file formats that can be licensed for royalty-free commercial use. They supplied the MIDI files used on the enclosed CD.

Contact:

One Gadsden Station  
Havana, FL 32333  
Telephone: (904) 574-3400

**Altura Software, Inc.**

Developer of software to compile and run Viewer titles on Macintosh computers. Also offers a similar package for Windows Help files and libraries that assist in porting applications in either direction between Windows and Macintosh environments.

Contact:

510 Lighthouse Avenue, Suite 5  
Pacific Grove, CA 93950  
Telephone: (408) 655-8005  
Fax: (408) 655-9663

