About WinHelp macro

About()

The **About** macro displays Windows Help's About dialog box.

Parameters

This macro does not take any parameters.

Comments

Use of this macro in secondary windows is not recommended.

See Also

Help Macros

AddAccelerator WinHelp macro

AddAccelerator(*key, shift-state,* "*macro*")

The **AddAccelerator** macro assigns a Help macro to an accelerator key (or key combination) so that the macro is carried out when the user presses the accelerator key(s).

Parameter	Description
key	Specifies the Windows virtual-key value. See the **Virtual key codes** topic for a list of virtual-key codes that may be used for this parameter.
shift-state	Specifies the combination of ALT, SHIFT, and CTRL keys to be used with the accelerator. This parameter may be one of the following values:

Value	Meaning
0	None
1	SHIFT
2	CTRL
3	SHIFT+CTRL
4	ALT
5	ALT+SHIFT
6	ALT+CTRL
7	SHIFT+ALT+CTRL

macro	Specifies the Help macro or macro string executed when the user presses the accelerator key(s). The macro must appear in quotation marks. Multiple macros in a string must be separated by semicolons.

Comments

The **AddAccelerator** macro can be abbreviated as **AA**.

Example

The following macro executes the Windows Clock program when the user presses ALT+SHIFT+**CONTROL**+F4:

```
AddAccelerator(0x73, 7, "ExecProgram(`clock.exe', 1)")
```

Related Topics

RemoveAccelerator

See Also

Help Macros

Annotate WinHelp macro

Annotate()

The **Annotate** macro displays the Annotation dialog box from the Edit menu.

Parameters

This macro does not take any parameters.

Comments

Use of this macro in secondary windows is not recommended.

See Also

Help Macros

AppendItem WinHelp macro

AppendItem("*menu-id*", "*item-id*", "*item-name*", "*macro*")

The **AppendItem** macro appends a menu item to the end of a menu created with the **InsertMenu** macro.

Parameter	Description
menu-id	Specifies the name used in the **InsertMenu** macro used to create the menu. This name must appear in quotation marks. The new item is appended to this menu.
item-id	Specifies the name that Windows Help uses internally to identify the menu item. This name must appear in quotation marks. This name is used by the **DisableItem** or **DeleteItem** macros.
item-name	Specifies the name that Windows Help displays on the menu for the item. This name must appear in quotation marks. Within the quotation marks, place an ampersand (&) before the character used for the macro's accelerator key.
macro	Specifies one or more macros that are to be executed when the user chooses the menu item. The macro must appear in quotation marks. Multiple macros in a string must be separated by semicolons (;).

Comments

Windows Help ignores this macro if it is executed in a secondary window.

If the keyboard accelerator conflicts with other menu access keys, Windows Help displays the error message "Unable to add item" and ignores the macro.

Example

The following macro appends a menu item labeled "Tools" to a pop-up menu that has an identifier "IDM_TLS". Choosing the menu item causes a jump to a topic with the context string "tpc1" in the TLS.HLP file:

```
AppendItem("IDM_BKS", "IDM_TLS", "&Tools", "JI(`tls.hlp', `tpc1')")
```

Related Topics

DeleteItem
DisableItem
InsertMenu
Help Macros

Back WinHelp macro

Back()

The **Back** macro displays the previous topic in the history list. The history list is a list of the last 40 topics the user has displayed since starting Windows Help.

Parameters

This macro does not take any parameters.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

If the **Back** macro is executed when the Back list is empty, Windows Help takes no action.

See Also

Help Macros

BookmarkDefine WinHelp macro

BookmarkDefine()

The **BookmarkDefine** macro displays the Define dialog from the Bookmark menu.

Parameters

This macro does not take any parameters.

Comments

Use of this macro in secondary windows is not recommended.

If the **BookmarkDefine** macro is executed from a pop-up window, the bookmark is attached to the topic that invoked the pop-up window.

See Also

Help Macros

BookmarkMore WinHelp macro

BookmarkMore()

The **BookmarkMore** macro displays the More dialog from the Bookmark menu. The More command appears on the Bookmark menu if the menu lists more than nine bookmarks.

Parameters

This macro does not take any parameters.

Comments

Use of the macro in secondary windows is not recommended.

See Also

Help Macros

BrowseButtons WinHelp macro

BrowseButtons()

The **BrowseButtons** macro adds browse buttons to the button bar.

Parameters

This macro does not take any parameters.

Comments

Windows Help ignores this macro if it is executed from a secondary window.

If the **BrowseButtons** macro is used with one or more **CreateButton** macros in the **[CONFIG] section** of the project file, the order of the browse buttons on the Windows Help button bar is determined by the order of the **BrowseButtons** macro in relation to the other macros listed in the [CONFIG] section.

Example

The following macros in the project file cause the Clock button to appear immediately before the two browse buttons on the button bar:

```
[CONFIG]
CreateButton("&Clock", "ExecProgram(`clock', 0)")
BrowseButtons()
```

Related Topics

CreateButton

Help Macros

ChangeButtonBinding WinHelp macro

ChangeButtonBinding("*button-id*", "*button-macro*")

The **ChangeButtonBinding** macro assigns a Help macro to a Help button.

Parameter	Description
button-id	Specifies the identifier assigned to the button by the **CreateButton** macro or, for a standard Help button, one of the following predefined button identifiers:

ID	Description
BTN_CONTENTS	Contents
BTN_SEARCH	Search
BTN_BACK	Back
BTN_HISTORY	History
BTN_PREVIOUS	Browse previous
BTN_NEXT	Browse next

The button identifier must be enclosed in quotation marks.

button-macro	Specifies the Help macro executed when the user selects the button. The macro must be enclosed in quotation marks.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

The **ChangeButtonBinding** macro can be abbreviated as **CBB**.

Example

In the following macro, "conts" is the context string for the table of contents in the DICT.HLP file:

```
ChangeButtonBinding("btn_contents", "JumpId(`dict.hlp', `conts')")
```

See Also

Help Macros

ChangeItemBinding WinHelp macro

ChangeItemBinding("*item-id*", "*item-macro*")

The **ChangeItemBinding** macro assigns a Help macro to an item previously added to a Windows Help menu using the **AppendItem** macro.

Parameter	Description
item-id	Identifies the menu item appended by the **AppendItem** macro. The item identifier must be enclosed in quotation marks.
item-macro	Specifies the Help macro to execute when the user selects the item. The macro must be enclosed in quotation marks.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

The **ChangeItemBinding** macro can be abbreviated as **CIB**.

Example

The following macro changes the menu item identified by "time_item" so that it displays the Windows clock:

```
ChangeItemBinding("time_item", "ExecProgram(`clock', 0)")
```

CheckItem WinHelp macro

CheckItem("*item-id*")

The **CheckItem** macro places a check-mark beside a menu item.

Parameter	Description
item-id	Identifies the menu item to check. The item identifier must be enclosed in quotation marks.

Comments

The **CheckItem** macro can be abbreviated as **CI**.

Related Topics

UncheckItem

See Also

Help Macros

CloseWindow WinHelp macro

CloseWindow("*window-name*")

The **CloseWindow** macro closes either a secondary window or the main Help window.

Parameter	Description
window-name	Specifies the name of the window to close. The name "main" is reserved for the main Help window. For secondary windows, the window name is defined in the **[WINDOWS]** section of the project file. This name must be enclosed in quotation marks.

Example
The following macro closes the secondary window named "keys":

```
CloseWindow("keys")
```

See Also

Help Macros

Contents WinHelp macro

Contents()

The **Contents** macro displays the Contents topic in the current Help file. The Contents topic is defined by the **CONTENTS** option in the **[OPTIONS]** section of the project file. If the project file does not have a **CONTENTS** option, the Contents topic is the first topic of the first topic file specified in the project file.

CopyDialog WinHelp macro

CopyDialog()

The **CopyDialog** macro displays the Copy dialog from the Edit menu.

Comments

Use of this macro in secondary windows is not recommended.

See Also

Help Macros

CopyTopic WinHelp macro

CopyTopic()

The **CopyTopic** macro copies all the text in the currently displayed topic to the Clipboard.

Comments

Use of the macro in secondary windows is not recommended.

See Also

Help Macros

CreateButton WinHelp macro

CreateButton("*button-id*", "*name*", "*macro*")

The **CreateButton** macro adds a new button to the button bar.

Parameter	Description
button-id	Specifies the name that WinHelp uses internally to identify the button. This name must appear in quotation marks. Use this name in the **DisableButton** or **DestroyButton** macro if you want to remove or disable the button or in the **ChangeButtonBinding** if you want to change the Help macro that the button executes in certain topics.
name	Specifies the text that appears on the button. To make a letter in this text the mnemonic for the button, place an ampersand (&) before that letter. The button name is case-sensitive and can have up to 29 characters in it -- any additional characters are ignored.
macro	Specifies the Help macro or macro string executed when the user clicks on the button. Multiple macros in a macro string must be separated by semicolons.

Comments

Windows Help allows a maximum of 16 custom buttons. It allows a total of 22 buttons, including the standard Browse buttons, on the button bar.

If the **BrowseButtons** macro is used with one or more **CreateButton** macros in the project file, the buttons appear in the same order on the button bar as the macros appear in the project file.

Windows Help ignores this macro if it is executed in a secondary window.

The **CreateButton** macro can be abbreviated as **CB**.

Example

The following macro creates a new button labeled "Ideas" that jumps to the topic with the context string "dir" in the IDEAS.HLP file when clicked:

```
CreateButton("btn_ideas", "&Ideas", "JumpId(`ideas.hlp', `dir')")
```

Related Topics

DisableButton, **DestroyButton**, **ChangeButtonBinding**, **JumpId**

Help Macros

DeleteItem WinHelp macro

DeleteItem("*item-id*")

The **DeleteItem** macro removes a menu item that was added by using the **AppendItem** macro.

Parameter	Description
item-id	Specifies the item identifier used in the **AppendItem** macro. The item identifier must be enclosed in quotation marks.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

Example

The following macro removes the menu item "Tools" appended in the example for the **AppendItem** macro:

```
DeleteItem("IDM_TOOLS")
```

Related Topics

AppendItem

See Also

Help Macros

DeleteMark WinHelp macro

DeleteMark("*marker-text***")**

The **DeleteMark** macro removes a text marker added with the **SaveMark** macro.

Parameter	Description
marker-text	Specifies the text marker previously added by the **SaveMark** macro. The marker text must be enclosed in quotation marks.

Comments

If the marker does not exist when the **DeleteMark** macro is executed, Windows Help displays a "Topic not found" error message.

Example

The following macro removes the marker "Managing Memory" from a Help file:

```
DeleteMark("Managing Memory")
```

Related Topics

SaveMark

See Also

Help Macros

DestroyButton WinHelp macro

DestroyButton("*button-id*")

The **DestroyButton** macro removes a button added with the **CreateButton** macro.

Parameter	Description
button-id	Identifies a button previously created by the **CreateButton** macro. The button identifier must be enclosed in quotation marks.

Comments

The button identifier cannot be an identifier for one of the standard Help buttons. For a list of those identifiers, see the **ChangeButtonBinding** macro.

Windows Help ignores this macro if it is executed in a secondary window.

Related Topics

CreateButton, **ChangeButtonBinding**

See Also

Help Macros

DisableButton WinHelp macro

DisableButton("*button-id*")

The **DisableButton** macro grays out a button added with the **CreateButton** macro. This button cannot be used in the topic until an **EnableButton** macro is executed.

Parameter	Description
button-id	Specifies the identifier assigned to the button by the **CreateButton** macro. The button identifier must be enclosed in quotation marks.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

The **DisableButton** macro can be abbreviated as **DB**.

Related Topics

CreateButton, **EnableButton**

See Also

Help Macros

DisableItem WinHelp macro

DisableItem("*item-id*")

The **DisableItem** macro grays out a menu item added with the **AppendItem** macro. The menu item cannot be used in the topic until an **EnableItem** macro is executed.

Parameter	Description
item-id	Identifies a menu item previously appended with the **AppendItem** macro. The item identifier must be enclosed in quotation marks.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

The **DisableItem** macro can be abbreviated as **DI**.

Related Topics

AppendItem

See Also

Help Macros

EnableButton WinHelp macro

EnableButton("*button-id*")

The **EnableButton** macro re-enables a button disabled with the **DisableButton** macro.

Parameter	Description
button-id	Specifies the identifier assigned to the button by the **CreateButton** macro. The button identifier must be enclosed in quotation marks.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

The **EnableButton** macro can be abbreviated as **EB**.

Related Topics

CreateButton, **DisableButton**

See Also

Help Macros

EnableItem WinHelp macro

EnableItem("*item-id*")

The **EnableItem** macro re-enables a menu item disabled with the **DisableItem** macro.

Parameter	Description
item-id	Specifies the identifier assigned to the menu item by the **AppendItem** macro. The item identifier must be enclosed in quotation marks.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

The **EnableItem** macro can be abbreviated as **EI**.

Related Topics

AppendItem, **DisableItem**

See Also

Help Macros

ExecProgram WinHelp macro

ExecProgram("*command-line*", *display-state*)

The **ExecProgram** macro executes a Windows application.

Parameter	Description
command-line	Specifies the command line for the application to be executed. The command line must be enclosed in quotation marks. Windows Help searches for this application in the current directory, followed by the Windows directory, the user's path, and the directory of the currently viewed Help file.
display-state	Specifies a value indicating how the application is shown when executed. It may be one of the following values:

Value	Meaning
0	Normal
1	Minimized
2	Maximized

Comments

The **ExecProgram** macro can be abbreviated as **EP**.

The backslash character should not be used to escape double quotation-mark characters in macros. Instead, you can enclose the command line in single quotation marks and omit the backslash for the double quotation marks, as shown in the following:

```
`command "string as parameter"'
```

Note that the first single quotation mark must be an open quote and the last single quotation mark must be a close quote.

Example

The following example executes the Clock application. The application is minimized when it starts:

```
ExecProgram(`clock.exe', 1)
```

See Also

Help Macros

Exit WinHelp macro

Exit()

The **Exit** macro exits the Windows Help application. It has the same effect as selecting Exit from the File menu.

Parameters

This macro does not take any parameters.

See Also

Help Macros

FileOpen WinHelp macro

FileOpen()

The **FileOpen** macro displays the Open dialog box from the File menu.

Parameters

This macro does not take any parameters.

Comments

Use of the macro in secondary windows is not recommended.

See Also

Help Macros

FocusWindow WinHelp macro

FocusWindow("*window-name*")

The **FocusWindow** macro changes the focus to the specified window, either the main Help window or a secondary window.

Parameter	Description
window-name	Specifies the name of the window to receive the focus. The name "main" is reserved for the main Help window. For secondary windows, the window name is defined in the **[WINDOWS] section** of the project file. This name must be enclosed in quotation marks.

Comments

This macro is ignored if the specified window does not exist.

Example

The following macro changes the focus to the secondary window "keys":

```
FocusWindow("keys")
```

See Also

Help Macros

GoToMark WinHelp macro

GoToMark("*marker-text*")

The **GoToMark** macro jumps to a marker set with the **SaveMark** macro.

Parameter	Description
marker-text	Specifies a text marker previously defined by using the **SaveMark** macro.

Example
The following macros jumps to the marker "Managing Memory".

```
GoToMark("Managing Memory")
```

Related Topics

SaveMark

See Also

Help Macros

HelpOn WinHelp macro

HelpOn()

The **HelpOn** macro displays the Help file for the Windows Help application. The macro carries out the same action as choosing the How to Use Help command on the Help menu.

Parameters

This macro does not take any parameters.

See Also

Help Macros

HelpOnTop WinHelp macro

HelpOnTop()

The **HelpOnTop** macro toggles the on-top state of Windows Help. It is equivalent to checking or unchecking the Always On Top command in the Help menu.

Parameters

This macro does not take any parameters.

Comments

Windows Help does not provide a macro to check the current state of the Always On Top command. It is up to the user to determine whether the macro should be used to change the state of the command.

See Also

Help Macros

History WinHelp macro

History()

The **History** macro displays the history list, which shows the last 40 topics the user has viewed since opening a Help file in Windows Help. It has the same effect as choosing the History button.

Parameters

This macro does not take any parameters.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

See Also

Help Macros

IfThen WinHelp macro

IfThen(**IsMark**("*marker-text*"), "*macro*")

The **IfThen** macro executes a Help macro if a given marker exists. It uses the **IsMark** macro to make the test.

Parameter	Description
marker-text	Specifies a text marker previously created by using the **SaveMark** macro. The marker must be enclosed in quotation marks.
macro	Specifies a Help macro or macro string to be executed if the marker exists. Multiple macros in a macro string must be separated by semicolons.

Example

The following macro jumps to the topic with context string "man_mem" if a marker named "Managing Memory" has been set by the **SaveMark** macro:

```
IfThen(IsMark("Managing Memory"), "JI(`trb.hlp', `man_mem')")
```

Related Topics

IsMark, **SaveMark**

See Also

Help Macros

IfThenElse WinHelp macro

IfThenElse(**IsMark**("*marker-text*"), "*macro1*", "*macro2*")

The **IfThenElse** macro executes one of two Help macros depending on whether or not a marker exists. It uses the **IsMark** macro to make the test.

Parameter	Description
marker-text	Specifies a text marker previously created by using the **IsMark** macro. The marker must be enclosed in quotation marks.
macro1	Specifies a Help macro or macro string to be executed if the marker exits. Multiple macros in either macro string must be separated by semicolons.
macro2	Specifies a Help macro or macro string to be executed if the marker does not exit. Multiple macros in either macro string must be separated by semicolons.

Example

The following macro jumps to the topic with context string "mem" if a marker named "Memory" has been set by the **SaveMark** macro. If the marker does not exist, it jumps to the next topic in the browse sequence.

```
IfThenElse(IsMark("Memory"), "JI(`trb.hlp', `mem')", "Next()")
```

Related Topics

IfThen, **IsMark**, **SaveMark**

See Also

Help Macros

InsertItem WinHelp macro

InsertItem("*menu-id*", "*item-id*", "*item-name*", "*macro*", *position*)

The **InsertItem** macro inserts a menu item at a given position on an existing menu. The menu can be either one you create with the **InsertMenu** macro or one of the standard Windows Help menus.

Parameter	Description
menu-id	Identifies either a standard Windows Help menu or a menu previously created by using the **InsertMenu** macro. For a standard menu, this parameter can be one of the following:

Name	Menu
MNU_FILE	File
MNU_EDIT	Edit
MNU_BOOKMARK	Bookmark menu
MNU_HELPON	Help

For other menus, this parameter must be the name used with the **InsertMenu** macro. In all cases, the menu identifier must be enclosed in quotation marks. The new item is inserted into this menu.

item-id	Specifies the name that Windows Help uses internally to identify the menu item. The item identifier must be enclosed in quotation marks.
item-name	Specifies the name Windows Help displays in the menu for the item. This name is case-sensitive and must be enclosed in quotation marks. An ampersand (&) before a character in the name identifies it as the item's keyboard access key.
macro	Specifies a Help macro or macro string to be executed when the user chooses the menu item. The macro must be enclosed in quotation marks. Multiple macros in a string must be separated by semicolons (;).
position	Specifies the position of the menu item in the menu. It must be an integer value. Position 0 is the first or topmost position in the menu.

Comments

The *item-id* parameter can be used in a subsequent **DisableItem** or **DeleteItem** macro to remove or disable the item or to change the operations that the item performs in certain topics.

Windows Help ignores this macro if it is executed in a secondary window.

The specified keyboard access keys must be unique. If a key conflicts with other menu access keys, Windows Help displays the error message "Unable to add item" and ignores the macro.

Example

The following macro inserts a menu item labeled "Tools" as the third item on a menu that has an identifier "MNU_BKS". Selecting the menu item causes a jump to a topic with the context string "tls1" in the TLS.HLP file:

```
InsertItem("mnu_bks", "m_tls", "&Tools", "JI(`tls.hlp', `tls1')", 3)
```

Related Topics

InsertMenu

See Also

Help Macros

InsertMenu WinHelp macro

InsertMenu("*menu-id*", "*menu-name*", *menu-position*)

The **InsertMenu** function inserts a new menu in the Windows Help menu bar.

Parameter	Description
menu-id	Specifies the name that Windows Help uses internally to identify the menu. The menu identifier must be enclosed in quotation marks. This identifier can be used in the **AppendItem** macro to add macros to the menu.
menu-name	Specifies the name that Windows Help displays on the menu bar. This name must be enclosed in quotation marks. An ampersand (&) before a character in the name identifies it as the menu's keyboard access key.
menu-position	Specifies the position on the menu bar of the new menu name. This parameter must be an integer number. Positions are numbered from left to right, with position 0 the left-most menu.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

Example

The following macro adds a menu named "Utilities" to the Windows Help application. The label "Utilities" appears as the fourth item on the Windows Help menu bar. The user presses U with the ALT key to open the menu.

```
InsertMenu("IDM_UTIL", "&Utilities", 3)
```

Related Topics

AppendItem, **InsertItem**

See Also

Help Macros

IsMark WinHelp macro

IsMark("*marker-text*")

The **IsMark** macro tests whether or not a marker set by the **SaveMark** macro exists. It is used as a parameter to the conditional macros **IfThen** and **IfThenElse**. The **IsMark** macro returns nonzero if the mark exists or zero if it does not.

Parameter	Description
marker-text	Specifies a text marker previous created using the **SaveMark** macro.

Comments

The **Not** macro can be used to reverse the results of the **IsMark** macro.

Example

The following macro jumps to the topic with the context string "man_mem" if a marker named "Managing Memory" has been set by the **SaveMark** macro:

```
IfThen(IsMark("Managing Memory"), "JI(`trb.hlp', `man_mem')")
```

Related Topics

IfThen, **IfThenElse**, **Not**

See Also

Help Macros

JumpContents WinHelp macro

JumpContents("*filename*")

The **JumpContents** macro jumps to the Contents topic of a specified file in the Help file. The Contents topic is indicated by the CONTENTS option entry in the **[OPTIONS] section** of project file. If the CONTENTS option is not specified, Windows Help jumps to the first topic in the Help file.

Parameter	Description
filename	Specifies the name of the destination file for the jump. The filename must be enclosed in quotation marks. If Windows Help cannot find this file, it displays an error message and does not perform the jump.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

Example

The following macro jumps to the Contents topic of the PROGMAN.HLP file:

```
JumpContents("PROGMAN.HLP")
```

Related Topics

JumpContext

See Also

Help Macros

JumpContext WinHelp macro

JumpContext("*filename*", *context-number*)

Parameter	Description
filename	Specifies the name of the destination file for the jump. The filename must be enclosed in quotation marks. If Windows Help cannot find this file, it displays an error message and does not perform the jump.
context-number	Specifies the context number of the topic in the destination file. The context number must be defined in the [MAP] section of the project file. If the context number is not valid, Windows Help jumps to the Contents topic or to the first topic in the file instead and displays an error message.

Comments

The **JumpContext** macro can be abbreviated as **JC**.

Example

The following macro jumps to the topic mapped to the context number 801 in the PROGMAN.HLP file:

```
JumpContext("PROGMAN.HLP", 801)
```

Related Topics

JumpContents

See Also

Help Macros

JumpHelpOn WinHelp macro

JumpHelpOn()

The **JumpHelpOn** macro jumps to the Contents topic of the How to Use Help file. The How To Use Help file is either the default WINHELP.HLP file shipped with Windows 3.1 or the Help file designated by the **SetHelpOnFile** macro in the **[CONFIG] section** of the project file.

Parameters

This macro does not take any parameters.

Comments

If Windows Help cannot find the specified Help file, it displays an error message and does not perform the jump.

Example

The following macro jumps to the Contents topic of the designated How to Use Help file:

```
JumpHelpOn()
```

See Also

Help Macros

JumpId WinHelp macro

JumpId("*filename*", "*context-string*")

The **JumpId** macro jumps to the topic with the specified context string in the Help file.

Parameter	Description
filename	Specifies the name of the Help file containing the context string. The filename must be enclosed in quotation marks. If Windows Help does not find this file, it displays an error message and does not perform the jump.
context-string	Context string of the topic in the destination file. The context string must be enclosed in quotation marks. If the context string does not exist, Windows Help jumps to the Contents topic for that file instead.

Comments

The **JumpId** macro may be abbreviated as **JI**.

Example

The following macro jumps to a topic with "second_topic" as its context string in the SECOND.HLP file:

```
JI("second.hlp", "second_topic")
```

See Also

Help Macros

JumpKeyword WinHelp macro

JumpKeyword("*filename*", "*keyword*")

The **JumpKeyword** macro loads the indicated Help file, searches through the K keyword table, and displays the first topic containing the index keyword specified in the macro.

Parameter	Description
filename	Specifies the name of the Help file containing the desired keyword table. The filename must be enclosed in quotation marks. If this file does not exist, Windows Help displays an error message and does not perform the jump.
keyword	Specifies the keyword that the macro searches for. The keyword must be enclosed in quotation marks. If Windows Help finds more than one match, it displays the first matched topic. If it does not find any matches, it displays a "Not a keyword" message and displays the Contents topic of the destination file instead.

Comments

The **JumpKeyword** macro can be abbreviated as **JK**.

Example

The following macro displays the first topic that has "hands" as an index keyword in the CLOCK.HLP file:

```
JumpKeyword("clock.hlp", "hands")
```

See Also

Help Macros

Next WinHelp macro

Next()

The **Next** macro displays the next topic in the browse sequence for the Help file.

Parameters

This macro does not take any parameters.

Comments

If the currently displayed topic is the last topic of a browse sequence, this macro does nothing.

Windows Help ignores this macro if it is executed in a secondary window.

See Also

Help Macros

Not WinHelp macro

Not(**IsMark**("*marker-text*"))

The **Not** macro reverses the result (nonzero or zero) returned by the **IsMark** macro. It is used along with the **IsMark** macro as a parameter to the conditional macros **IfThen** and **IfThenElse**.

Parameter	Description
marker-text	Specifies a text marker previously created by using the **SaveMark** macro. The marker text must be enclosed in quotation marks.

Example

The following macro jumps to the topic with the context string "mem1" if a marker named "Memory" has not been set by the **SaveMark** macro:

```
IfThen(Not(IsMark("Memory")), "JI(`trb.hlp', `mem1')")
```

Related Topics

IfThen, **IfThenElse**, **IsMark**

See Also

Help Macros

PopupContext WinHelp macro

PopupContext("*filename*", *context-number*)

The **PopupContext** macro displays in a pop-up window the topic identified by a specific context number.

Parameter	Description
filename	Specifies the name of the file that contains the topic to be displayed. The filename must be enclosed in quotation marks. If Windows Help cannot find this file, it displays an error message.
context number	Specifies the context number of the topic to be displayed. The context number must be specified in the [MAP] section of the project file. If the context number is not valid, Windows Help displays the Contents topic or the first topic in the file instead.

Comments

The **PopupContext** macro can be abbreviated as **PC**.

Example

The following macro displays in a pop-up window the topic mapped to the context number 801 in the PROGMAN.HLP file:

```
PopupContext("progman.hlp", 801)
```

Related Topics

PopupId

See Also

Help Macros

PopupId WinHelp macro

PopupId("*filename*", "*context-string*")

The **PopupId** macro displays a topic from a specified file in a pop-up window.

Parameter	Description
filename	Specifies the name of the file containing the pop-up window topic. The filename must be enclosed in quotation marks. If this file does not exist, Windows Help displays a warning.
context-string	Specifies the context string of the topic in the destination file. If the requested context string does not exist, Windows Help displays the Contents topic or the first topic in the file.

Comments

The **PopupId** macro can be abbreviated as **PI**.

Example

The following macro displays a pop-up window with context string "second_topic" from the SECOND.HLP file:

```
PopupId("second.hlp", "second_topic")
```

Related Topics

PopupContext

See Also

Help Macros

PositionWindow WinHelp macro

PositionWindow(*x, y, width, height, state,* "*name*")

The **PositionWindow** macro sets the size and position of a window.

Parameter	Description
x	Specifies the x-coordinate, in help units, of the upper-left corner of the window. Windows Help always assumes the screen (regardless of resolution) is 1024 help units wide. For example, if the x-coordinate is 512, the left edge of the Help window is in the middle of the screen.
y	Specifies the y-coordinate, in help units, of the upper-left corner of the window. Windows Help always assumes the screen (regardless of resolution) is 1024 help units high. For example, if the y-coordinate is 512, the top edge of the Help window is in the middle of the screen.
width	Specifies the default width, in help units, of the window.
height	Specifies the default height, in help units, of the window.
state	Specifies how the window is sized. This parameter can be one of the following values:

Value	Meaning
0	Normal size
1	Maximized

If the parameter is 1, Windows Help ignores the *x, y, width,* and *height* parameters.

name	Specifies the name of the window to position. The name "main" is reserved for the main Help window. For secondary windows, the window name must be defined in the [WINDOWS] section of the project file. This name must be enclosed in quotation marks.

Comments

If the window to be positioned does not exist, Windows Help ignores the macro.

The **PositionWindow** macro can be abbreviated as **PW**.

Example

The following macro positions the secondary window "Samples" in the upper-left corner (100, 100) with a width and height of 500 (in help units):

```
PositionWindow(100, 100, 500, 500, 0, "Samples")
```

See Also

Help Macros

Prev WinHelp macro

Prev()

The **Prev** macro displays the previous topic in the browse sequence for the Help file. If the currently displayed topic is the first topic of a browse sequence, this macro does nothing.

Parameters

This macro does not take any parameters.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

See Also

Help Macros

Print WinHelp macro

Print()

The **Print** macro sends the currently displayed topic to the printer. It should be used only to print topics in windows other than the main Help window (for example, topics in a secondary window).

Parameters

This macro does not take any parameters.

Related Topics

PrinterSetup

Help Macros

PrinterSetup WinHelp macro

PrinterSetup()

The **PrinterSetup** macro displays the Printer Setup dialog box from the File menu.

Parameters

This macro does not take any parameters.

Comments

Use of the macro in secondary windows is not recommended.

Related Topics

Print

Help Macros

RegisterRoutine WinHelp macro

RegisterRoutine("*DLL-name*", "*function-name*", "*format-spec*")

The **RegisterRoutine** macro registers a function within a dynamic-link library (DLL). Registered functions can be used in macro footnotes in topic files or in the **[CONFIG]** section of the project file, the same as standard Help macros.

Parameter	Description
DLL-name	Specifies the filename of the DLL. The filename must be enclosed in quotation marks. If Windows Help cannot find the library, it displays an error message.
function-name	Specifies the name of the function to execute in the designated DLL.
format-spec	Specifies a string indicating the formats of parameters passed to the function. The format string must be enclosed in quotation marks. Characters in the string represent C parameter types:

Character	Description
u	**unsigned short** (**WORD**)
U	**unsigned long** (**DWORD**)
i	**short int**
I	**int**
s	**near char** * (PSTR)
S	**far char** * (**LPSTR**)
v	**void**

If the function is used as a Help macro, Windows Help makes sure the macro parameters match the parameter types given in this macro.

Comments

The **RegisterRoutine** macro can be abbreviated as **RR**.

Example

The following call registers a routine named PlayAudio in a DLL, MMLIB.DLL. PlayAudio takes arguments of the **far char** *, **int**, and **unsigned long** types:

```
RegisterRoutine("MMLIB", "PlayAudio", "SIU")
```

See Also

Help Macros

RemoveAccelerator WinHelp macro

RemoveAccelerator(*key, shift-state*)

The **RemoveAccelerator** macro removes the assignment of a Help macro to an accelerator key (or key combination). These assignments are made by using the **AddAccelerator** macro.

Parameter	Description
key	Specifies the Windows virtual-key value. See the **Virtual key codes** topic in the Windows SDKfor a list of virtual-key codes that may be used for this parameter.
shift-state	Specifies the combination of ALT, SHIFT, and CTRL keys that were used with the accelerator. This parameter may be one of the following values:

Value	Meaning
0	None
1	SHIFT
2	CTRL
3	SHIFT+CTRL
4	ALT
5	ALT+SHIFT
6	ALT+CTRL
7	SHIFT+ALT+CTRL

Comments

The **RemoveAccelerator** macro can be abbreviated as **RA**. No error occurs when this macro is used with an accelerator for which a macro was not defined.

Example

The following macro disassociates a macro from the ALT+SHIFT+**CONTROL**+F4 key combination:

```
RemoveAccelerator(0x73, 7)
```

Related Topics

AddAccelerator

See Also

Help Macros

SaveMark WinHelp macro

SaveMark("*marker-text*")

The **SaveMark** macro saves the location of the currently displayed topic and file and associates a text marker with that location. The **GotoMark** macro can then be used to jump to this location.

Parameter	Description
marker-text	Specifies the text marker to be used to identify the topic location. This text must be enclosed in quotation marks, and it must be unique. If the same text is used for more than one marker, the most recently entered marker is used.

Comments

A text marker can be used with the **GotoMark**, **DeleteMark**, **IfThen**, and **IfThenElse** macros.

If the user exits Windows Help, all text markers are deleted.

Example

The following macro saves the marker "Managing Memory" in the current topic:

```
SaveMark("Managing Memory")
```

Related Topics

DeleteMark, **GotoMark**, **IfThen**, **IfThenElse**, **IsMark**, **Not**

See Also

Help Macros

Search WinHelp macro

Search()

The **Search** macro displays the dialog for the Search button, which allows users to search for topics using keywords defined by the K footnote character.

Parameters

This macro does not take any parameters.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

See Also

Help Macros

SetContents WinHelp macro

SetContents("*filename*", *context-number*)

The **SetContents** macro designates a specific topic as the Contents topic in the specified Help file.

Parameter	Description
filename	Specifies the name of the Help file that contains the Contents topic. The filename must be enclosed in quotation marks. If Windows Help cannot find this file, it displays an error message and does not perform the jump.
context number	Specifies the context number of the topic in the specified file. The context number must be defined in the [MAP] section of the project file. If the context number is not valid, Windows Help displays an error message.

Example

The following example sets the topic mapped to the context number 801 in the PROGMAN.HLP file as the Contents topic. After executing this macro, clicking the Contents button will cause a jump to the topic specified by the *context-number* parameter:

```
SetContents("PROGMAN.HLP", 801)
```

See Also

Help Macros

SetHelpOnFile WinHelp macro

SetHelpOnFile("*filename*")

Parameter	Description
filename	Specifies the name of the replacement How to Use Help file. The filename must be enclosed in quotation marks. If Windows Help cannot find this file, it displays an error message.

Comments

If this macro appears in a topic in the Help file, the replacement file is set after execution of the macro. If this macro appears in the **[CONFIG] section** of the project file, the replacement file is set when the help file is opened.

Example

The following macro sets the Using Help file to MYHELP.HLP:

```
SetHelpOnFile("myhelp.hlp")
```

See Also

Help Macros

UncheckItem WinHelp macro

UncheckItem("*item-id*")

The **UncheckItem** macro removes the check mark from a menu item.

Parameter	Description
item-id	Identifies the menu item to uncheck. The item identifier must be enclosed in quotation marks.

Comments

The **UncheckItem** macro can be abbreviated **UI**.

Related Topics

CheckItem

See Also

Help Macros

Help Macros- Alphabetical

Macro Reference, Categorical

About WinHelp macro	Displays the About dialog box
AddAccelerator WinHelp macro	Assigns a macro to an accelerator key
Annotate WinHelp macro	Displays Annotation dialog box
AppendItem WinHelp macro	Appends a menu item
Back WinHelp macro	Displays previous topic in the history list
BookmarkDefine WinHelp macro	Displays the Define dialog box
BookmarkMore WinHelp macro	Displays the More dialog box
BrowseButtons WinHelp macro	Adds browse buttons
ChangeButtonBinding WinHelp macro	Assigns a macro to a button
ChangeItemBinding WinHelp macro	Assigns a macro to a menu item
CheckItem WinHelp macro	Checks a menu item
CloseWindow WinHelp macro	Closes a window
Contents WinHelp macro	Displays the Contents topic
CopyDialog WinHelp macro	Displays the Copy dialog box
CopyTopic WinHelp macro	Copies current topic to the clipboard
CreateButton WinHelp macro	Adds a new button to the button bar
DeleteItem WinHelp macro	Removes a menu item
DeleteMark WinHelp macro	Deletes a text marker
DestroyButton WinHelp macro	Removes a button from the button bar
DisableButton WinHelp macro	Disables a button
DisableItem WinHelp macro	Disables a menu item
EnableButton WinHelp macro	Enables a button
EnableItem WinHelp macro	Enables a menu item
ExecProgram WinHelp macro	Executes a program
Exit WinHelp macro	Exits WinHelp
FileOpen WinHelp macro	Displays the Open dialog box
FocusWindow WinHelp macro	Changes the focus window
GoToMark WinHelp macro	Jumps to a marker
HelpOn WinHelp macro	Displays the Help on Using topic
HelpOnTop WinHelp macro	Toggles on-top state of help
History WinHelp macro	Displays the history list
IfThen WinHelp macro	Executes macro if marker exists
IfThenElse WinHelp macro	Executes one of two macros if marker exists
InsertItem WinHelp macro	Inserts a menu item
InsertMenu WinHelp macro	Inserts a new menu
IsMark WinHelp macro	Tests if a marker is set
JumpContents WinHelp macro	Jumps to the Contents topic
JumpContext WinHelp macro	Jumps to the specified context
JumpHelpOn WinHelp macro	Jumps to Using Help file
JumpId WinHelp macro	Jumps to the specified topic
JumpKeyword WinHelp macro	Jumps to the topic containing the keyword
Next WinHelp macro	Displays the next topic in the browse sequence
Not WinHelp macro	Reverses the IsMark macro
PopupContext WinHelp macro	Displays a topic in a popup window
PopupId WinHelp macro	Displays topic in a popup window
PositionWindow WinHelp macro	Sets the size and position of a window
Prev WinHelp macro	Displays previous topic in browse sequence
Print WinHelp macro	Prints the current topic
PrinterSetup WinHelp macro	Displays the Printer Setup dialog box
RegisterRoutine WinHelp macro	Registers a DLL function
RemoveAccelerator WinHelp macro	Assigns a macro to an accelerator key
SaveMark WinHelp macro	Saves a marker

Search WinHelp macro	Displays the Search dialog box
SetContents WinHelp macro	Sets the Contents topic
SetHelpOnFile WinHelp macro	Sets the Using Help help file
UncheckItem WinHelp macro	Unchecks a menu item

Virtual Key Codes

The following table shows the symbolic constant names, hexadecimal values, and keyboard equivalents for the virtual-key codes used by the Windows operating system version 3.1. The codes are listed in numeric order.

Symbolic name	Value (in hex)	Mouse or keyboard equivalent
VK_LBUTTON	01	Left mouse button
VK_RBUTTON	02	Right mouse button
VK_CANCEL	03	Used for control-break processing
VK_MBUTTON	04	Middle mouse button (three-button mouse)
--	05-07	Undefined
VK_BACK	08	BACKSPACE key
VK_TAB	09	TAB key
--	0A0B	Undefined
VK_CLEAR	0C	CLEAR key
VK_RETURN	0D	ENTER key
--	0E0F	Undefined
VK_SHIFT	10	SHIFT key
VK_CONTROL	11	CTRL key
VK_MENU	12	ALT key
VK_PAUSE	13	PAUSE key
VK_CAPITAL	14	CAPS LOCK key
--	15-19	Reserved for Kanji systems
--	1A	Undefined
VK_ESCAPE	1B	ESC key
--	1C1F	Reserved for Kanji systems
VK_SPACE	20	SPACEBAR
VK_PRIOR	21	PAGE UP key
VK_NEXT	22	PAGE DOWN key
VK_END	23	END key
VK_HOME	24	HOME key
VK_LEFT	25	LEFT ARROW key
VK_UP	26	UP ARROW key
VK_RIGHT	27	RIGHT ARROW key
VK_DOWN	28	DOWN ARROW key
VK_SELECT	29	SELECT key
--	2A	OEM specific
VK_EXECUTE	2B	EXECUTE key
VK_SNAPSHOT	2C	PRINT SCREEN key for Windows 3.0 and later
VK_INSERT	2D	INS key
VK_DELETE	2E	DEL key
VK_HELP	2F	HELP key
VK_0	30	0 key
VK_1	31	1 key
VK_2	32	2 key

VK_3	33	3 key
VK_4	34	4 key
VK_5	35	5 key
VK_6	36	6 key
VK_7	37	7 key
VK_8	38	8 key
VK_9	39	9 key
--	3A40	Undefined
VK_A	41	A key
VK_B	42	B key
VK_C	43	C key
VK_D	44	D key
VK_E	45	E key
VK_F	46	F key
VK_G	47	G key
VK_H	48	H key
VK_I	49	I key
VK_J	4A	J key
VK_K	4B	K key
VK_L	4C	L key
VK_M	4D	M key
VK_N	4E	N key
VK_O	4F	O key
VK_P	50	P key
VK_Q	51	Q key
VK_R	52	R key
VK_S	53	S key
VK_T	54	T key
VK_U	55	U key
VK_V	56	V key
VK_W	57	W key
VK_X	58	X key
VK_Y	59	Y key
VK_Z	5A	Z key
--	5B5F	Undefined
VK_NUMPAD0	60	Numeric keypad 0 key
VK_NUMPAD1	61	Numeric keypad 1 key
VK_NUMPAD2	62	Numeric keypad 2 key
VK_NUMPAD3	63	Numeric keypad 3 key
VK_NUMPAD4	64	Numeric keypad 4 key
VK_NUMPAD5	65	Numeric keypad 5 key
VK_NUMPAD6	66	Numeric keypad 6 key
VK_NUMPAD7	67	Numeric keypad 7 key
VK_NUMPAD8	68	Numeric keypad 8 key
VK_NUMPAD9	69	Numeric keypad 9 key

VK_MULTIPLY	6A	Multiply key
VK_ADD	6B	Add key
VK_SEPARATOR	6C	Separator key
VK_SUBTRACT	6D	Subtract key
VK_DECIMAL	6E	Decimal key
VK_DIVIDE	6F	Divide key
VK_F1	70	F1 key
VK_F2	71	F2 key
VK_F3	72	F3 key
VK_F4	73	F4 key
VK_F5	74	F5 key
VK_F6	75	F6 key
VK_F7	76	F7 key
VK_F8	77	F8 key
VK_F9	78	F9 key
VK_F10	79	F10 key
VK_F11	7A	F11 key
VK_F12	7B	F12 key
VK_F13	7C	F13 key
VK_F14	7D	F14 key
VK_F15	7E	F15 key
VK_F16	7F	F16 key
VK_F17	80H	F17 key
VK_F18	81H	F18 key
VK_F19	82H	F19 key
VK_F20	83H	F20 key
VK_F21	84H	F21 key
VK_F22	85H	F22 key
VK_F23	86H	F23 key
VK_F24	87H	F24 key
--	88-8F	Unassigned
VK_NUMLOCK	90	NUM LOCK key
VK_SCROLL	91	SCROLL LOCK key
--	92B9	Unassigned
--	BAC0	OEM specific
--	C1DA	Unassigned
--	DBE4	OEM specific
--	E5	Unassigned
--	E6	OEM specific
--	E7E8	Unassigned
--	E9F5	OEM specific
--	F6FE	Unassigned

[ALIAS] Section

[ALIAS]
context_string = alias
.
.
.

The **[ALIAS]** section assigns one or more context strings to the same topic alias. This section is optional.

Parameter	Description
context_string	Specifies the context string that identifies a particular topic. This context string may be used in a hotspot or in the **[MAP] section** to refer to a particular topic.
alias	Specifies the alternative string or alias name. This string is used in the **\footnote** statement. An alias string has the same form and follows the same conventions as the topic context string. That is, it is not case-sensitive and may contain the alphabetic characters A through Z, the numeric characters 0 through 9, and the period and underscore characters.

Comments

Because context strings must be unique for each topic and cannot be used for any other topic in the Help project, the **[ALIAS]** section provides a way to delete or combine help topics without recoding your files. For example, if you create a topic that replaces information in three other topics, you could manually search through your files for invalid cross-references to the deleted topics. The easier approach, however, would be to use the **[ALIAS]** section to assign the name of the new topic to the deleted topics.

The **[ALIAS]** section can also be used when your application has multiple context identifiers for one help topic. This situation occurs in context-sensitive Help.

Alias names can be used in a **[MAP]** section, but only if the **[ALIAS]** section precedes the **[MAP]** section.

Example

The following example creates several aliases:

```
[ALIAS]
sm_key=key_shrtcuts
cc_key=key_shrtcuts
st_key=key_shrtcuts     ; combined into Keyboard Shortcuts topic
clskey=us_dlog_bxs
maakey=us_dlog_bxs      ; covered in Using Dialog Boxes topic.
chk_key=dlogprts
drp_key=dlogprts
lst_key=dlogprts
opt_key=dlogprts
tbx_key=dlogprts        ; combined into Parts of Dialog Box topic.
frmtxt=edittxt
wrptxt=edittxt
seltxt=edittxt          ; covered in Editing Text topic.
```

See Also

[MAP]
HPJ Statements

[BAGGAGE] Section

[BAGGAGE]
filename
.
.
.

The **[BAGGAGE]** section lists files (typically multimedia elements) that the Help Compiler stores within the help file's internal file system. Windows Help can access data files stored in the help file more efficiently than it can access files in the normal MS-DOS file system, since it doesn't have to read the file allocation table from CD-ROM.

Parameter	Description
filename	Specifies the full path of a file. If a file cannot be found, the compiler reports an error.

Comments

A maximum of 1,000 files can be stored as baggage files.

If a file is listed in the **[BAGGAGE]** section, you must use or write a dynamic-link library that uses Windows Help to read these files from the help file.

See Also
ROOT
HPJ Statements

Changes

[BITMAPS] Section

[BITMAPS]
filename
.
.
.

The **[BITMAPS]** section specifies the names and locations of the bitmap files specified in the **bmc**, **bml**, and **bmr** statements.

Parameter	Description
filename	Specifies the full path of a bitmap file. If a file cannot be found, the compiler reports an error.

Comments
For Windows 3.1, the **[BITMAPS]** section is not required if the bitmaps are located in the Help project directory or if the path containing the bitmaps is listed in the **BMROOT** or **ROOT** option. If the project file does not include either of these options, each bitmap filename must be listed in the **[BITMAPS]** section of the project file.

Example
The following example specifies three bitmap files:

```
[BITMAPS]
BMP01.BMP
BMP02.BMP
BMP03.BMP
```

See Also

BMROOT
ROOT
HPJ Statements

Changes for Windows 3.1

For Windows 3.1, the **[BITMAPS]** section is not required if the bitmaps are located in the Help project directory or if the path containing the bitmaps is listed in the **BMROOT** or **ROOT** option. For Windows 3.0, all bitmaps used in the help file must be placed in the **[BITMAPS]** section.

BMROOT Option

BMROOT = *path*[**,** *path*]...

The **BMROOT** option specifies the directory containing the bitmap files specified in the **bmc**, **bml**, and **bmr** statements.

Parameter	Description
path	Specifies a drive and full path.

Comments

If the project file has a **BMROOT** option, you do not need to list the bitmap files in the **[BITMAPS]** section.

If the project file does not have a **BMROOT** option, the Help compiler looks for bitmaps in the directories specified by the **ROOT** option. If the project file does not have a **ROOT** option or if the **ROOT** option does not specify the directory containing the bitmap files, the filename for each bitmap must be specified in the **[BITMAPS]** section.

Example

The following example specifies that bitmaps are in the \HELP\BMP directory on drive C: and the \GRAPHICS\ART directory on drive D:

```
[OPTIONS]
BMROOT=C:\HELP\BMP, D:\GRAPHICS\ART
```

See Also

[BITMAPS]
[OPTIONS]Section
ROOT

BUILD Option

BUILD = *expression*

The **BUILD** option specifies which topics containing build tags are included in a build. The **BUILD** option does not apply to topics that do not contain build tags.

A topic contains a build tag if it contains a build-tag **\footnote** statement. Topics without build tags are always compiled, regardless of the current build expression.

Parameter	Description
expression	Specifies the build expression. This parameter consists of a combination of build tags (specified in the **[BUILDTAGS]** section) and the following operators:

Operator	Description
~	Applies the NOT operator to a single tag. The Help compiler compiles a topic only if the tag is *not* present. This operator has the highest precedence; the compiler applies it before any other operator.
&	Combines two tags by using the AND operator. The Help compiler compiles a topic only if it contains both tags. The compiler applies this operator only after the ~ operator has been applied.
|	Combines two tags by using the OR operator. The Help compiler compiles a topic if it has at least one tag. This operator has the lowest precedence; the compiler applies it only after all other operators have been applied.

Parentheses may be used to override operator precedence. Expressions enclosed in parentheses are always evaluated first.

Comments

Only one **BUILD** option can be given per project file.

The Help compiler evaluates all build expressions from left to right, using the specified precedence rules.

Example

The following examples assume that the **[BUILDTAGS]** section in the project file defines the build tags DEMO, MASTER, and TEST_BUILD. Although the following examples show several **BUILD** options on consecutive lines, only one **BUILD** option per project file is allowed.

```
BUILD = DEMO             ; compile topics that have the DEMO tag
BUILD = DEMO & MASTER    ; compile topics with both DEMO and MASTER
BUILD = DEMO | MASTER    ; compile topics with either DEMO or MASTER
BUILD = ~DEMO            ; compile topics that do not have DEMO
BUILD = (DEMO | MASTER) & TEST_BUILD
                         ; compile topics that have TEST_BUILD and
                         ; either DEMO or MASTER
```

See Also

[BUILDTAGS]
[OPTIONS]Section

[BUILDTAGS] Section

[BUILDTAGS]
tag
.
.
.

The **[BUILDTAGS]** section defines the build tags for the help file. The Help compiler uses these tags to determine which topics to include when building the help file.

This section is used in conjunction with the build-tag **\footnote** statements. These **\footnote** statements associate a build tag with a given topic. If the build tag is also defined in the **[BUILDTAGS]** section, the Help compiler compiles the topic; otherwise, it ignores the topic.

Parameter	Description
tag	Specifies a build tag consisting of any combination of characters except spaces. The Help compiler strips any space characters from the tag. Also, the compiler treats uppercase and lowercase characters as the same characters (that is, it is case-insensitive).

Comments
The **[BUILDTAGS]** section is optional. If given, it can contain up to 30 build tags.

Example
The following example shows the form of the **[BUILDTAGS]** section in a sample project file:

```
[BUILDTAGS]
DEMO            ; topics to include in demo build
MASTER          ; topics to include in master help file
DEBUGBUILD      ; topics to include in debugging build
TESTBUILD       ; topics to include in a mini-build for testing
```

See Also

BUILD
HPJ Statements

CITATION Option

CITATION = *citation*

The **CITATION** option places a custom citation in the About dialog box of Windows Help. Windows Help displays the citation immediately below the copyright notice.

Parameter	Description
citation	Specifies the citation. The notice can be any combination of characters; its length must be in the range 35 through 75 characters.

See Also

COPYRIGHT
[OPTIONS]Section

COMPRESS Option

COMPRESS = *compression-level*

The **COMPRESS** option specifies the level of compression to be used when building the help file. Compression levels indicate either no compression, medium compression (approximately 40%), or high compression (approximately 50%).

Parameter	Description
compression-level	Specifies the level of compression. This parameter can be one of the following values:

Value	Meaning
0	No compression
1	High compression
FALSE	No compression
HIGH	High compression
MEDIUM	Medium compression
NO	No compression
TRUE	High compression
YES	High compression

Comments

Depending on the degree of compression requested, the build uses either block compression or a combination of block and key-phrase compression. Block compression compresses the topic data into predefined units known as blocks. Key-phrase compression combines repeated phrases found within the source file(s). The compiler creates a phrase-table file with the .PH extension if one does not already exist. If the compiler finds a file with the .PH extension, it uses that file for the current compilation. Because the .PH file speeds up the compression process when little text has changed since the last compilation, you might want to keep the phrase file if you compile the same Help file several times with compression. However, you will get maximum compression if you delete the .PH file before starting each build.

See Also

[OPTIONS]Section

[CONFIG] Section

[CONFIG]
macro
.
.
.

The **[CONFIG]** section contains one or more macros that carry out actions, such as enabling browse buttons and registering dynamic-link library (DLL) functions. Windows Help executes the macros when it opens the help file.

Parameter	Description
macro	Specifies a Windows Help macro.

Comments
The **[CONFIG]** section may include any number of lines. Each line of the **[CONFIG]** section may be up to 254 characters long.

Example
The following example registers a DLL, creates a button, enables the browse buttons, and sets the name of the help file containing information about how to use Help:

```
[CONFIG]
RegisterRoutine("bmp","HDisplayBmp","USSS")
RegisterRoutine("bmp","CopyBmp", "v=USS")
CreateButton("btn_up", "&Up", "JumpContents(`HOME.HLP')")
BrowseButtons()
SetHelpOnFile("APPHELP.HLP")
```

See Also

HPJ Statements

CONTENTS Option

CONTENTS = *context-string*

The **CONTENTS** option identifies the context string of the highest-level or Contents topic. This topic is usually a table of contents or index within the help file. Windows Help displays the Contents topic whenever the user clicks the Contents button.

Parameter	Description
context-string	Specifies the context string of a topic in the help file. The string can be any combination of characters, except spaces, and must also be specified in a context-string **\footnote** statement in some topic in the help file.

Comments

If the **[OPTIONS]** section does not include a **CONTENTS** option, the compiler assumes that the Contents topic is the first topic encountered in the first listed topic file in the **[FILES]** section of the project file.

The **CONTENTS** option is equivalent to the **INDEX** option that was available in Windows version 3.0.

Example

The following example sets the topic containing the context string "main_contents" as the Contents topic:

```
CONTENTS=main_contents
```

See Also

[FILES]
[OPTIONS]Section

COPYRIGHT Option

COPYRIGHT = *copyright-notice*

The **COPYRIGHT** option places a custom copyright notice in the About dialog box of Windows Help. Windows Help displays the notice immediately below the copyright notice.

Parameter	Description
copyright-notice	Specifies the copyright notice. The notice can be any combination of characters; its length must be in the range 35 through 75 characters.

Comments

The copyright notice is also appended to topics that are copied to the clipboard, unless it is replaced by using the **CITATION** option.

See Also

CITATION
[OPTIONS]Section

ERRORLOG Option

ERRORLOG = *error-filename*

The **ERRORLOG** option directs the Help compiler to write all error messages to the specified file. The compiler also displays the error messages on the screen.

Parameter	Description
error-filename	Specifies the name of the file to receive the error messages. This parameter can be a full or partial path if the error file should be written to a directory other than the project root directory.

Example

The following example writes all errors during the build to the HLPBUGS.TXT file in the Help project root directory.

```
ERRORLOG=HLPBUGS.TXT
```

See Also

[OPTIONS]Section

[FILES] Section

[FILES]
filename
.
.
.

The **[FILES]** section lists all topic files used to build the help file. Every project file requires a **[FILES]** section.

Parameter	Description
filename	Specifies the full or partial path of a topic file. If a partial path is given, the Help compiler uses the directories specified by the **ROOT** option to construct a full path. If a file cannot be found, the compiler reports an error.

Comments
The **#include** directive can also be used in the **[FILES]** section to specify the topic files indirectly by designating a file that contains a list of the topic files.

Example
The following example specifies four topic files:

```
[FILES]
rtftxt\COMMANDS.RTF
rtftxt\HOWTO.RTF
rtftxt\KEYS.RTF
rtftxt\GLOSSARY.RTF
```

The following example uses the **#include** directive to specify the topic files indirectly. In this case, the file RTFFILES.H must be in the project file (the Help compiler does not use the INCLUDE environment variable to search for files).

```
[FILES]
#include <rtffiles.h>
```

See Also

ROOT
HPJ Statements

FORCEFONT Option

FORCEFONT = *fontname*

The **FORCEFONT** option forces the specified font to be substituted for all requested fonts. The option is used to create help files that can be viewed on systems that do not have all fonts available.

Parameter	Description
fontname	Specifies the name of an available font. Font names must be spelled the same as they are in the Fonts dialog box in Control Panel. Font names cannot exceed 20 characters. If an invalid font name is given, the Help compiler uses the MS Sans Serif font as the default.

See Also

[OPTIONS]Section

ICON Option

ICON = *icon-file*

The **ICON** option identifies the icon file to display when the user minimizes Windows Help.

Parameter	Description
icon-file	Specifies the name of the icon file. This file must have the standard Windows icon-file format.

See Also

[OPTIONS]Section

LANGUAGE Option

LANGUAGE = *language-name*

The **LANGUAGE** option sets the sorting order for keywords in the Search dialog box.

Parameter	Description
language-name	Specifies the language on which to base sorting. This parameter can be the following:

Value	Meaning
scandanavian	Sets the sorting order to the Scandavanian-language order.

Comments

The default sorting order is the English-language order.

See Also

[OPTIONS]Section

[MAP] Section

[MAP]
context-string context-number
.
.
.

The **[MAP]** section associates context strings (or aliases) with context numbers for context-sensitive Help. The context number corresponds to a value the parent application passes to Windows Help in order to display a particular topic. This section is optional.

Parameter	Description
context-string	Specifies the context string of a topic in the help file. The string can be any combination of characters, except spaces, and must also be specified in a context-string **\footnote** statement in some topic in the help file.
context-number	Specifies the context number to associate with the context string. The number can be in either decimal or standard C hexadecimal format. Only one context number may be assigned to a context string or alias. Assigning the same number to more than one context string generates a compiler error. At least one space must separate the context number from the context string.

Comments
You can define the context strings listed in the **[MAP]** section either in a help topic or in the **[ALIAS]** section. The compiler generates a warning message if a context string appearing in the **[MAP]** section is not defined in any of the topic files or in the **[ALIAS]** section.

If you use an alias name, the **[ALIAS]** section must precede the **[MAP]** section in the Help project file.

The **[MAP]** section supports two additional statements for specifying context strings and their associated context numbers. The first statement has the following form:

#define *context-string context-number*

The *context-string* and *context-number* parameters are as described in the Parameters section.

The second statement has the following form:

#include "*filename*"

The *filename* parameter, which can be enclosed in either double quotation marks or angle brackets(<>), specifies the name of a file containing one or more **#define** statements. The file may contain additional **#include** statements as well, but files may not be nested in this way more than five deep.

Example
The following example assigns hexadecimal context numbers to the context strings:

```
[MAP]
Edit_Window     0x0001
Control_Menu    0x0002
Maximize_Icon   0x0003
Minimize_Icon   0x0004
Split_Bar       0x0005
Scroll_Bar      0x0006
```

```
Title_Bar       0x0007
Window_Border   0x0008
```

See Also

[ALIAS]
HPJ Statements

MAPFONTSIZE Option

MAPFONTSIZE = *m:p*

The **MAPFONTSIZE** option maps font sizes specified in topic files to different sizes when they are displayed in the Help window. This option is especially useful if there is a significant size difference between the authoring display and the intended user display.

Parameter	Description
m	Specifies the size of the source font. This parameter is either a single point size or a range of point sizes. A range of point sizes consists of the low and high point sizes separated by a hyphen (-). If a range is specified, all fonts in the range are changed to the size specified by the *p* parameter.
p	Specifies the size of the desired font for the help file.

Comments

Although the **[OPTIONS]** section can contain up to five font ranges, only one font size or range is allowed with each **MAPFONTSIZE** statement. If more than one **MAPFONTSIZE** statement is included, the source font size or range specified in subsequent statements cannot overlap previous mappings.

Example

The following examples illustrate the use of the **MAPFONTSIZE** option:

```
MAPFONTSIZE=8:12      ; display all 8-pt. fonts as 12-pt.
MAPFONTSIZE=12-24:16  ; display fonts from 12 to 24 pts. as 16 pts.
```

See Also

[OPTIONS]Section

MULTIKEY Option

MULTIKEY = *footnote-character*

The **MULTIKEY** option specifies the footnote character to use for an alternative keyword table. This option is intended to be used in conjunction with topic files that contain **\ footnote** statements for alternative keywords.

Parameter	Description
footnote-character	Specifies the case-sensitive letter to be used for the keyword footnote.

Comments

Since keyword footnotes are case-sensitive, you should limit your keyword-table footnotes to one case, usually uppercase. If an uppercase letter is specified, the compiler will not include footnotes with the lowercase form of the same letter in the keyword table.

You may use any alphanumeric character for a keyword table except *K* and *k,* which are reserved for Help's standard keyword table. There is an absolute limit of five keyword tables, including the standard table. However, depending upon system configuration and the structure of your Help system, a practical limit of only two or three tables may be more realistic. If the compiler cannot create an additional keyword table, the additional table is ignored in the build.

Example

The following example illustrates how to enable the letter *L* for a keyword-table footnote:

```
MULTIKEY=L
```

See Also

[OPTIONS]Section

OLDKEYPHRASE Option

OLDKEYPHRASE = *onoff*

The **OLDKEYPHRASE** option specifies whether an existing key-phrase file should be used to build the help file.

Parameter	Description
onoff	Specifies whether the existing file should be used. This parameter can be one of the following values:

Value	Meaning
0	Recreate the file
1	Use the existing file
FALSE	Recreate the file
NO	Recreate the file
OFF	Recreate the file
ON	Use the existing file
TRUE	Use the existing file
YES	Use the existing file

See Also

[OPTIONS]Section

OPTCDROM Option

OPTCDROM = *yesvalue*

The **OPTCDROM** option optimizes a help file for display on CD-ROM by aligning topic files on predefined block boundaries.

Parameter	Description
yesvalue	Specifies that the file should be optimized for CD-ROM. This parameter can be any of the following values: **YES** **TRUE** **1** **ON**

See Also

[OPTIONS]Section

[OPTIONS] Section

[OPTIONS]
option
.
.
.

The **[OPTIONS]** section includes options that control how a help file is built and what feedback the build process displays. If this section is included in the project file, it should be the first section listed, so that the options will apply during the entire build process.

Parameter	Description
option	Specifies one of the following project-file options:

Option	Description
BMROOT	Specifies the directory containing the bitmap files named in the **bmc**, **bml**, and **bmr** statements in topic files. This option is new for Windows 3.1.
BUILD	Specifies which topics to include in the build.
CITATION	Specifies a string that is appended to topics that are copied from Windows Help instead of the **COPYRIGHT** string. This option is new for Windows 3.1.
COMPRESS	Specifies the type of compression to use during the build.
CONTENTS	Specifies the context string of the Contents topic for a help file. This option is new for Windows 3.1.
COPYRIGHT	Adds a unique copyright message for the help file to the About dialog box. This option is new for Windows 3.1.
ERRORLOG	Puts compilation errors in a file during the build. This option is new for Windows 3.1.
FORCEFONT	Forces all authored fonts in the topic files to appear in a different font when displayed in the Help window.
ICON	Specifies the icon file to be displayed when the help file is minimized. This option is new for Windows 3.1.
LANGUAGE	Specifies a different sorting order for help files authored in a Scandanavian language.
MAPFONTSIZE	Maps a font size in the topic file to a different font size in the compiled help file.
MULTIKEY	Specifies an alternative keyword table to use for mapping topics.
OLDKEYPHRASE	Specifies whether the compiler should use the existing key-phrase table or create a new one during the build. This option is new for Windows 3.1.
OPTCDROM	Optimizes the help file for CD-ROM use. This option is new for Windows 3.1.
REPORT	Controls the display of messages during the build process.
ROOT	Specifies the directories containing the topic and data files listed in the project file.
TITLE	Specifies the text displayed in the title bar of the Help

	window when the file is open.
WARNING	Specifies the level of error-message reporting the compiler is to display during the build.

Comments

These options can appear in any order within the **[OPTIONS]** section. The **[OPTIONS]** section is not required.

See Also

HPJ Statements

REPORT Option

REPORT = ON

The **REPORT** option displays messages on the screen during the build. These messages indicate when the Help compiler is performing the different phases of the build, including compiling the file, resolving jumps, and verifying browse sequences.

See Also

[OPTIONS]Section
WARNING

ROOT Option

ROOT = *pathname*[**,** *pathname*]...

The **ROOT** option specifies the directories where the Help compiler looks for files listed in the project file.

Parameter	Description
pathname	Specifies either a drive and full path or a relative path from the project directory. If the project file has a **ROOT** option, all relative paths in the project file refer to one of these paths. If the project file does not have a **ROOT** option, all paths are relative to the directory containing the project file.

Comments

If the project file does not have a **BMROOT** option, the compiler looks in the directories specified in the **ROOT** option to find bitmaps positioned by using the **bmc**, **bml**, and **bmr** statements. If none of these directories contains these bitmaps, the bitmap filenames must be listed in the **[BITMAPS]** section of the project file.

Example

The following example specifies that the project root directory is C:\WINHELP\HELPDIR and is found on drive C:

```
[OPTIONS]
ROOT=C:\WINHELP\HELPDIR
```

Given this root directory, if the **[FILES]** section contains the entry TOPICS\FILE.RTF, the full path for the topic file is C:\WINHELP\HELPDIR\TOPICS\FILE.RTF.

See Also

[BITMAPS]
BMROOT
[OPTIONS]Section

TITLE Option

TITLE = *titlename*

The **TITLE** option sets the title for the help file. Windows Help displays the title in its title bar whenever it displays the help file.

Parameter	Description
titlename	Specifies the title displayed in the Windows Help title bar. The title must not exceed 50 characters.

Comments

If no title is specified by using the **TITLE** option, Windows Help displays the title Windows Help in the title bar.

Example

The following example sets the help-file title to **ABC** Help.

```
[OPTIONS]
TITLE=ABC Help
```

See Also

[OPTIONS]Section

WARNING Option

WARNING = *level*

The **WARNING** option specifies the amount of debugging information the Help compiler is to report.

Parameter	Description
level	Specifies the warning level. This parameter may be one of the following values:

Value	Meaning
1	Report only the most severe errors.
2	Report an intermediate number of errors.
3	Report all errors and warnings.

Example

The following example specifies an intermediate level of error reporting:

```
[OPTIONS]
WARNING=2
```

See Also

[OPTIONS] Section
REPORT

[WINDOWS] Section

[WINDOWS]
type = "*caption*", (*x, y, width, height*), *sizing,*
(*clientRGB*), (*nonscrollRGB*), (*fTop*)
.
.
.

The **[WINDOWS]** section defines the size, location, and colors for the primary Help window and any secondary-window types used in a help file.

The secondary windows defined in this section are intended to be used with Windows applications that specify secondary windows when calling the **WinHelp** function.

Parameter	Description
type	Specifies the type of window that uses the defined attributes. For the primary Help window, this parameter is **main**. For a secondary window, this parameter may be any unique name of up to 8 characters. Any jumps that display a topic in a secondary window give this type name as part of the jump.
caption	Specifies the title for a secondary window. Windows Help places the title in the title bar of the window. To set the title for the primary Help window, use the **TITLE** option in the **[OPTIONS]** section.
x	Specifies the x-coordinate, in help units, of the window's upper-left corner. Windows Help always assumes the screen is 1024 help units wide, regardless of resolution. For example, if the x-coordinate is 512, the left edge of the Help window is in the middle of the screen.
y	Specifies the y-coordinate, in help units, of the window's upper-left corner. Windows Help always assumes the screen is 1024 help units high, regardless of resolution. For example, if the x-coordinate is 512, the top edge of the Help window is in the middle of the screen.
width	Specifies the default width, in help units, for a secondary window.
height	Specifies the default height, in help units, for a secondary window.
sizing	Specifies the relative size of a secondary window when Windows Help first opens the window. This parameter can be one of the following values:

Value	Meaning
0	Set the window to the size specified by the *x, y, width,* and *height* parameters.
1	Maximize the window; ignore the *x, y, width,* and *height* parameters.

clientRGB	Specifies the background color of the window. This parameter is an **RGB** color value consisting of three 8-bit hexadecimal numbers enclosed in parentheses and separated by commas. If this parameter is not given, Windows Help uses the default window color specified by Control Panel.
nonscrollRGB	Specifies the background color of the non-scrolling region (if any) in the Help window. This parameter is an **RGB** color value consisting of three 8-bit hexadecimal numbers enclosed in parentheses and separated by commas. If this parameter is not given, Windows Help uses the default window color specified by Control Panel.
fTop	Specifies whether the secondary window is displayed on top of all other windows. When this parameter is 1, the window is diplayed over all windows that do not also use this attribute. Otherwise, it should be zero.

This parameter is optional.

Example

The following example defines two windows, the main window and a secondary window named "picture". The main-window definition sets the background color of non-scrolling regions in the main Help window to (128, 0, 128) but leaves several other values empty (for which Windows Help will supply its own default values). The secondary-window definition sets the caption to "Samples" and sets the width and height of the window to about one-quarter of the width and height of the screen. The background colors for the window and non-scrolling region are (0, 255, 255) and (255, 0, 0), respectively. The *sizing* parameter for both the main and secondary windows is zero.

```
[WINDOWS]
main=, (, , , ), 0, (, , ), (128, 0, 128)
picture = "Samples", (123,123,256,256), 0, (0,255,255), (255,0,0)
```

See Also

[OPTIONS]Section
TITLE
HPJ Statements

HPJ Statements

[ALIAS] Section	Assigns context strings to a topic alias
[BAGGAGE] Section	Lists files to add to the Help file
[BITMAPS] Section	Specifies the names of bitmap files
BMROOT Option	Specifies the directory containing bitmaps
BUILD Option	Specifies which topics to build
[BUILDTAGS] Section	Specifies valid build tags
CITATION Option	Inserts a citation string in the About dialog box
COMPRESS Option	Sets the level of compression for the help file
[CONFIG] Section	Specifies the Help file configuration
CONTENTS Option	Specifies the context string of the contents topic
COPYRIGHT Option	Inserts a copyright string in the About dialog box
ERRORLOG Option	Specifies the file to receive error messages
[FILES] Section	Specifies the topic files
FORCEFONT Option	Sets the Help file font
ICON Option	Specifies the Windows Help icon
LANGUAGE Option	Sets the sort-ordering for the keyword list
[MAP] Section	Associates context strings with context numbers
MAPFONTSIZE Option	Maps font sizes for the Help file
MULTIKEY Option	Specifies the footnote for alternate keywords
OLDKEYPHRASE Option	Specifies whether to use old phrase files
OPTCDROM Option	Optimizes help file for display on CD-ROM
[OPTIONS] Section	Contains options that control the Help compiler
REPORT Option	Displays build message during compilation
ROOT Option	Specifies the directories containing topic and data files
TITLE Option	Specifies the Help file title
WARNING Option	Specifies the warning level for error messages
[WINDOWS] Section	Contains definitions for Help windows

RTF Tokens

\ansi	Specifies the ANSI character set
\b	Starts bold text
\bin	Specifies binary picture data
bmc	Displays a bitmap or metafile in text
bml	Displays a bitmap or metafile at the left margin
bmr	Displays a bitmap or metafile at the right margin
\box	Draws a box
\brdrb	Draws a bottom border
\brdrbar	Draws a vertical bar
\brdrdb	Sets double-lined borders
\brdrdot	Sets dotted border
\brdrl	Draws a left border
\brdrr	Draws a right border
\brdrs	Sets standard borders
\brdrt	Draws a top border
\brdrth	Sets thick borders
\cell	Marks end of table cell
\cellx	Sets the position of a cell's right edge
\cf	Sets the foreground color
\colortbl	Creates the color table
\deff	Sets default font
emc	Allows DLL to paint window in text
eml	Allows DLL to paint window at left margin
emr	Allows DLL to paint window at right margin
\f	Sets the font
\fi	Sets the first-line indent
\fldrslt	Result of a field
\fonttbl	Creates the font table
\footnote	Defines topic-specific information
\fs	Sets the font size
\'	Inserts a character by value
\i	Starts italic text
\intbl	Marks paragraph as in table
\keep	Makes text non-wrapping
\keepn	Creates a non-scrolling region
\li	Sets the left indent
\line	Breaks the current line
\mac	Sets the Apple MacIntosh character set
\page	Ends current topic
\par	Marks the end of a paragraph
\pard	Restores default paragraph properties
\pc	Sets the PC character set
\pich	Specifies the picture height

\pichgoal	Specifies the desired picture height
\picscalex	Specifies the horizontal scaling value
\picscaley	Specifies the vertical scaling value
\pict	Creates a picture
\picw	Specifies the picture width
\picwgoal	Specifies the desired picture width
\plain	Restores default character properties
\qc	Centers text
\ql	Aligns text left
\qr	Aligns text right
\ri	Sets the right indent
\row	Marks end of a table row
\rtf	Specifies the RTF version
\sa	Sets the spacing after a paragraph
\sb	Sets space before
\scaps	Starts small capitals
\sect	Marks the end of a section and paragraph
\sl	Sets the spacing between lines
\strike	Creates a hotspot
\tab	Inserts a tab character
\tqc	Tabs and centers text
\tqr	Tabs and aligns text right
\trgaph	Sets space between text columns in a table
\trleft	Sets left margin for the first cell
\trowd	Sets table defaults
\trqc	Sets relative column widths
\trql	Left-aligns table row
\tx	Sets a tab stop
\ul	Creates a link to a pop-up topic
\uldb	Creates a hot spot
\v	Creates a link to a topic
\wbitmap	Specifies a Windows bitmap
\wbmbitspixel	Specifies the number of bits per pixel
\wbmplanes	Specifies the number of planes
\wbmwidthbytes	Specifies the bitmap width in bytes
\wmetafile	Specifies a Windows metafile

\ansi RTF statement

\ansi

The **\ansi** statement sets the American National Standards Institute (ANSI) character set. The Windows character set is essentially equivalent to the ANSI character set.

See Also

\mac
\pc

\b RTF statement

\b

The **\b** statement starts bold text. The statement applies to all subsequent text up to the next **\plain** or **\b0** statement.

Comments

No **\plain** or **\b0** statement is required if the **\b** statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

The **\b0** statement was first supported in the Help Compiler version 3.1.

Example

The following example sets "Note" to bold:

```
{\b Note}  Setting the Auto option frees novice users from
determining their system configurations.
```

See Also

\i
\plain
\scaps

\bin RTF statement

\bin*n*

The **\bin** statement indicates the start of binary picture data. The Help compiler interprets subsequent bytes in the file as binary data. This statement is used in conjunction with the **\pict** statement.

Parameter	Description
n	Specifies the number of bytes of binary data following the statement.

Comments

A single space character must separate the **\bin** statement from subsequent bytes. The Help Compiler assumes that all subsequent bytes, including linefeed and carriage return characters, are binary data. These bytes can have any value in the range 0 through 255. For this reason, the **\bin** statement is typically used in program-generated files only.

If the **\bin** statement is not given with a **\pict** statement, the default picture data format is hexadecimal.

See Also

\pict

bmc RTF statement

\{**bmc** *filename*\}

The **bmc** statement displays a specified bitmap or metafile in the current line of text. The statement positions the bitmap or metafile as if it were the next character in the line, aligning it on the base line and applying the current paragraph properties.

Parameter	Description
filename	Specifies the name of a file containing either a Windows bitmap, a placeable Windows metafile, a multiresolution bitmap, or a segmented-graphics bitmap.

Comments

Since the **bmc** statement is not a standard RTF statement, the Help Compiler relies on the opening and closing braces, including the backslashes (\), to distinguish the statement from regular text.

If a file containing a metafile is specified, the file must contain a placeable Windows metafile; the Help Compiler will not accept standard Windows metafiles. Furthermore, Windows Help sets the MM_ANISOTROPIC mode prior to displaying the metafile, so the placeable Windows metafile must either set the window origin and extents or set some other mapping mode.

See Also

bmr
bml
\wbitmap

bml RTF statement

\{**bml** *filename*\}

The **bml** statement displays a specified bitmap or metafile at the left margin of the Help window. The first line of subsequent text aligns with the upper-right corner of the image and subsequent lines wrap along the right edge of the image.

Parameter	Description
filename	Specifies the name of a file containing either a Windows bitmap, a placeable Windows metafile, a multiresolution bitmap, or a segmented-graphics bitmap.

Comments

Since the **bml** statement is not a standard RTF statement, the Help Compiler relies on the opening and closing braces, including the backslashes (\), to distinguish the statement from regular text.

If a file containing a metafile is specified, the file must contain a placeable Windows metafile; the Help Compiler will not accept standard Windows metafiles. Furthermore, Windows Help sets the MM_ANISOTROPIC mode prior to displaying the metafile, so the placeable Windows metafile must either set the window origin and extents or set some other mapping mode.

See Also

bmc
bmr
\wbitmap

bmr RTF statement

\{**bmr** *filename*\}

The **bmr** statement displays a specified bitmap or metafile at the right margin of the Help window. The first line of subsequent text aligns with the upper-left corner of the image and subsequent lines wrap along the left edge of the image.

Parameter	Description
filename	Specifies the name of a file containing either a Windows bitmap, a placeable Windows metafile, a multiresolution bitmap, or a segmented-graphics bitmap.

Comments

Since the **bmr** statement is not a standard RTF statement, the Help Compiler relies on the opening and closing braces, including the backslashes (\), to distinguish the statement from regular text.

If a file containing a metafile is specified, the file must contain a placeable Windows metafile; the Help compiler will not accept standard Windows metafiles. Furthermore, Windows Help sets the MM_ANISOTROPIC mode prior to displaying the metafile, so the placeable Windows metafile must either set the window origin and extents or set some other mapping mode.

See Also

bmc
bml
\wbitmap

\box RTF statement

\box

The **\box** statement draws a box around the current paragraph or picture. The statement applies to all subsequent paragraphs or pictures up to the next **\pard** statement.

Comments

For paragraphs, Windows Help uses the height of the paragraph, excluding space before or after the paragraph, as the height of the box. For pictures (as defined by **\pict** statements), Windows Help uses the specified height of the picture as the height of the box. For both paragraphs and pictures, the width of the box is equal to the space between the left and right indents.

Windows Help draws the box using the current border style.

Example

The following example draws a box around the paragraph:

```
\par \box
{\b Note}  Setting the Auto option frees novice users from
determining their system configurations.
\par \pard
```

See Also

\brdrb
\brdrl
\brdrr
\brdrt
\pard

\brdrb RTF statement

\brdrb

The **\brdrb** statement draws a border below the current paragraph or picture. The statement applies to all subsequent paragraphs or pictures up to the next **\pard** statement.

Comments

Windows Help draws the border using the current border style.

See Also

\box
\brdrbar
\brdrl
\brdrr
\brdrt
\pard

\brdrbar RTF statement

\brdrbar

The **\brdrbar** statement draws a vertical bar to the left of the current paragraph or picture. The statement applies to all subsequent paragraphs or pictures up to the next **\pard** statement.

Comments

Windows Help draws the border using the current border style.

In a print-based document, the **\brdrbar** statement draws the bar on the right side of paragraphs on odd-numbered pages, but on the left side of paragraphs on even-numbered pages.

See Also

\box
\brdrl
\brdrb
\brdrr
\brdrt
\pard

\brdrdb RTF statement

\brdrdb

The **\brdrdb** statement selects a double line for drawing borders. The selection applies to all subsequent paragraphs or pictures up to the next **\pard** statement.

See Also

\brdrdot
\brdrs
\brdrth
\pard

\brdrdot RTF statement

\brdrdot

The Help compiler ignores this statement.

See Also

\brdrs
\brdrth
\brdrdb
\pard

\brdrl RTF statement

\brdrl

The **\brdrl** statement draws a border to the left of the current paragraph or picture. The statement applies to all subsequent paragraphs or pictures up to the next **\pard** statement.

Comments

Windows Help draws the border using the current border style.

See Also

\box
\brdrb
\brdrbar
\brdrr
\brdrt
\pard

\brdrr RTF statement

\brdrr

The **\brdrr** statement draws a border to the right of the current paragraph or picture. The statement applies to all subsequent paragraphs or pictures up to the next **\pard** statement.

Comments

Windows Help draws the border using the current border style.

See Also

\box
\brdrb
\brdrbar
\brdrl
\brdrt
\pard

\brdrs RTF statement

\brdrs

The **\brdrs** statement selects a standard-width line for drawing borders. The selection applies to all subsequent paragraphs or pictures up to the next **\pard** statement.

See Also

\brdrdb
\brdrdot
\brdrth
\pard

\brdrt RTF statement

\brdrt

The **\brdrt** statement draws a border above the current paragraph or picture. The statement applies to all subsequent paragraphs or pictures up to the next **\pard** statement.

Comments

Windows Help draws the border using the current border style.

See Also

\box
\brdrb
\brdrbar
\brdrl
\brdrr
\pard

\brdrth RTF statement

\brdrth

The **\brdrth** statement selects a thick line for drawing borders. The selection applies to all subsequent paragraphs or pictures up to the next **\pard** statement.

See Also

\brdrdb
\brdrdot
\brdrs
\pard

\cell RTF statement (3.1)

\cell

The **\cell** statement marks the end of a cell in a table. A cell consists of all paragraphs from a preceding **\intbl** or **\cell** statement to the ending **\cell** statement. Windows Help formats and displays these paragraphs using the left and right margins of the cell and any current paragraph properties.

Comments

This statement was first supported in the Help Compiler version 3.1.

Example

The following example creates a two-column table. The second column contains three separate paragraphs, each having different paragraph properties:

```
\cellx2880\cellx5760
\intbl
Alignment\cell
\ql
Left-aligned
\par
\qc
Centered
\par
\qr
Right-aligned\cell
\row \pard
```

See Also

\cellx
\intbl
\row
\trgaph
\trleft
\trowd

\cellx RTF statement (3.1)

\cellx*n*

The **\cellx** statement sets the absolute position of the right edge of a table cell. One **\cellx** statement must be given for each cell in the table. The first **\cellx** statement applies to the left-most cell, the last to the right-most cell. For each **\cellx** statement, the specified position applies to the corresponding cell in each subsequent row of the table up to the next **\trowd** statement.

Parameter	Description
n	Specifies the position of the cell's right edge, in twips. The position is relative to the left edge of the Help window. It is not affected by the current indents.

Comments

A table consists of a grid of cells in columns and rows. Each cell has an explicitly defined right edge; the position of a cell's left edge is the same as the position of the right edge of the adjacent cell. For the left-most cell in a row, the left edge position is equal to the Help window's left margin position. Each cell has a left and right margin between which Windows Help aligns and wraps text. By default, the margin positions are equal to the left and right edges. The **\trgaph** and **\trleft** statements can be used to set different margins for all cells in a row.

This statement was first supported in the Help Compiler version 3.1.

Example

The following example creates a three-column table having two rows. The positions of the right edges of the three cells are 2, 4, and 6 inches, respectively:

```
\cellx2880\cellx5760\cellx8640
\intbl
Row 1 Cell 1\cell
Row 1 Cell 2\cell
Row 1 Cell 3\cell
\row
\intbl
Row 2 Cell 1\cell
Row 2 Cell 2\cell
Row 2 Cell 3\cell
\row \pard
```

See Also

\cell
\intbl
\row
\trgaph
\trleft
\trowd

\cf RTF statement

\cf*n*

The **\cf** statement sets the foreground color. The new color applies to all subsequent text up to the next **\plain** or **\cf** statement.

Parameter	Description
n	Specifies the color number to set as foreground. The number must be an integer number in the range 1 to the maximum number of colors specified in the color table for the Help file. If an invalid color number is specified, Windows Help uses the default foreground color.

Comments

No **\plain** or **\cf** statement is required if the **\cf** statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to the enclosed text only.

If the **\cf** statement is not given, the default foreground color is the text color set by Control Panel.

Example

The following example displays green text:

```
{\colortbl;\red0\green255\blue0;}
{\cf1 This text is green.}
```

See Also

\colortbl

\colortbl RTF statement

{**\colortbl**
\red*r***\green***g***\blue***b;*
.
.
.
}

The **\colortbl** statement creates a color table for the Help file. The color table consists of one or more color definitions. Each color definition consists of one **\red**, **\green**, and **\blue** statement specifying the amount of primary color to use to generate the final color. Each color definition must end with a semicolon (;).

Parameter	Description
r	Specifies the intensity of red in the color. It must be an integer in the range 0 through 255.
g	Specifies the intensity of green in the color. It must be an integer in the range 0 through 255.
b	Specifies the intensity of blue in the color. It must be an integer in the range 0 through 255.

Comments

Color definitions are implicitly numbered starting at zero. A color definition's implicit number can be used in the **\cf** statement to set the foreground color.

The default colors are the window-text and window-background colors set by Control Panel. To override the default colors, both a **\colortbl** statement and a **\cf** statement must be given.

Example

The following example creates a color table containing two color definitions. The first color definition is empty (only the semicolon is given), so color number 0 always represents the default color. The second definition specifies green; color number 1 can be used to display green text:

```
{\colortbl;\red0\green255\blue0;}
```

See Also

\cf

\deff RTF statement

\deff*n*

The **\deff** statement sets the default font number. Windows Help uses the number to set the default font whenever a **\plain** statement is given or an invalid font number is given in a **\f** statement.

Parameter	Description
n	Specifies the number of the font to be used as the default font. This parameter must be a valid font number as specified by the **\fonttbl** statement for the Help file.

Comments

If the **\deff** statement is not given, the default font number is zero.

See Also

\f
\fonttbl
\plain

emc RTF statement

\{**emc** *module, class, data* [, *dx, dy*]\}

The **emc** statement allows an external dynamic-link library to paint a window that is embedded in a Help topic. This statement displays the window in the current line of text. The statement positions the window as if it were the next character in the line, aligning it on the base line and applying the current paragraph properties.

Parameter	Description
module	Specifies the name of the dynamic-link library that paints the embedded window.
class	Specifies the name of the registered window class for the embedded window.
data	Specifies a string that is passed to the embedded window in its **WM_CREATE** message.
dx	Specifies the suggested width of the embedded window. This parameter is optional.
dy	Specifies the suggested height of the embedded window. This parameter is optional.

Comments

Since the **emc** statement is not a standard RTF statement, the Help Compiler relies on the opening and closing braces, including the backslashes (\), to distinguish the statement from regular text.

See Also

bmr
bml
bmc
eml
emr
\wbitmap

eml RTF statement

\{**eml** *module, class, data* [, *dx, dy*]\}

The **eml** statement allows an external dynamic-link library to paint a window that is embedded at the left margin in a Help topic. The first line of subsequent text aligns with the upper-right corner of the window and subsequent lines wrap along the right edge of the window.

Parameter	Description
module	Specifies the name of the dynamic-link library that paints the embedded window.
class	Specifies the name of the registered window class for the embedded window.
data	Specifies a string that is passed to the embedded window in its **WM_CREATE** message.
dx	Specifies the suggested width of the embedded window. This parameter is optional.
dy	Specifies the suggested height of the embedded window. This parameter is optional.

Comments

Since the **eml** statement is not a standard RTF statement, the Help Compiler relies on the opening and closing braces, including the backslashes (\), to distinguish the statement from regular text.

See Also

bmr
bml
bmc
emc
emr
\wbitmap

emr RTF statement

\{**emr** *module, class, data* [, *dx, dy*]\}

The **emr** statement allows an external dynamic-link library to paint a window that is embedded at the right margin in a Help topic. The first line of subsequent text aligns with the upper-left corner of the window and subsequent lines wrap along the left edge of the window.

Parameter	Description
module	Specifies the name of the dynamic-link library that paints the embedded window.
class	Specifies the name of the registered window class for the embedded window.
data	Specifies a string that is passed to the embedded window in its **WM_CREATE** message.
dx	Specifies the suggested width of the embedded window. This parameter is optional.
dy	Specifies the suggested height of the embedded window. This parameter is optional.

Comments

Since the **emr** statement is not a standard RTF statement, the Help Compiler relies on the opening and closing braces, including the backslashes (\), to distinguish the statement from regular text.

See Also

bmr
bml
bmc
emc
eml
\wbitmap

\f RTF statement

\f*n*

The **\f** statement sets the font. The new font applies to all subsequent text up to the next **\plain** or **\f** statement.

Parameter	Description
n	Specifies the font number. This parameter must be one of the integer font numbers defined in the font table for the Help file.

Comments

The **\f** statement does not set the point size of the font; use the **\fs** statement instead.

No **\plain** or **\f** statement is required if the **\f** statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

If the **\f** statement is not given, the default font is defined by the **\deff** statement (or is zero if no **\deff** statement is given).

Example

The following example uses the Arial font to display text:

```
{\fonttbl {\f0\fswiss Arial;}}
{\f0
This text illustrates the Arial font.}
\par
```

See Also

\deff
\fonttbl
\fs
\plain

\fi RTF statement

\fi*n*

The **\fi** statement sets the first-line indent for the paragraph. The new indent applies to the first line of each subsequent paragraph up to the next **\pard** statement or **\fi** statement. The first-line indent is always relative to the current left indent.

Parameter	Description
n	Specifies the indent, in twips. This parameter can be either a positive or negative number.

Comments

If the **\fi** statement is not given, the first-line indent is zero by default.

Example

The following example uses the first-line indent and a tab stop to make a numbered list:

```
\tx360\li360\fi-360
1
\tab
Insert the disk in drive A.
\par
2
\tab
Type a:setup and press the ENTER key.
\par
3
\tab
Follow the instructions on the screen.
\par \pard
```

See Also

\li
\pard

\fldrslt RTF statement

\fldrslt

The **\fldrslt** statement specifies the most recently calculated result of a field. The Help Compiler interprets the result as text and formats it using the current character and paragraph properties.

Comments

The Help compiler ignores all field statements except the **\fldrslt** statement. Any text associated with other field statements is ignored.

\fonttbl RTF statement

{**\fonttbl**
    {**\f***n**family font-name;*}
        .
        .
        .
}

The **\fonttbl** statement creates a font table for the Help file. The font table consists of one or more font definitions. Each definition consists of a font number, a font family, and a font name.

Parameter	Description
n	Specifies the font number. This parameter must be an integer. This number can be used in subsequent **\f** statements to set the current font to the specified font. In the font table, font numbers should start at zero and increase by one for each new font definition.
family	Specifies the font family. This parameter must be one of the following:

Value	Meaning
fnil	Unknown or default fonts (default)
froman	Roman, proportionally spaced serif fonts (for example, MS Serif and Palatino)
fswiss	Swiss, proportionally spaced sans serif fonts (for example, Swiss)
fmodern	Fixed-pitch serif and sans serif fonts (for example, Courier, Elite, and Pica)
fscript	Script fonts (for example, Cursive)
fdecor	Decorative fonts (for example, Old English and ITC Zapf Chancery)
ftech	Technical, symbol, and mathematical fonts (for example, Symbol)

font-name	Specifies the name of the font. This parameter should specify an available Windows font.

Comments

If a font with the specified name is not available, Windows Help chooses a font from the specified family. If no font from the given family exists, Windows Help chooses a font having the same character set as specified for the Help file.

The **\deff** statement sets the default font number for the Help file. The default font is set whenever the **\pard** statement is given.

See Also

\deff
\f
\fs
\pard

\footnote RTF statement

{*n*}{\footnote {*n*} *text*}

The **\footnote** statement defines topic-specific information, such as the topic's build tags, context string, title, browse number, keywords, and execution macros. Every topic must have a context string, at least, to give the user access to the topic through links.

Parameter	Description
n	Specifies the footnote character. It can be one of the following:

Value	Meaning
*	Specifies a build tag. The Help Compiler uses build tags to determine whether it should include the topic in the Help file. The *text* parameter can be any combination of characters but must not contain spaces. Uppercase and lowercase characters are treated as equivalent characters (case-insensitive). If a topic has build-tag statements, they must be the first statements in the topic. The Help Compiler checks a topic for build tags if the project file specifies a build expression using the **BUILD** option.
#	Specifies a context string. The *text* parameter can be any combination of letters and digits but must not contain spaces. Uppercase and lowercase characters are treated as equivalent characters (case-insensitive). The context string can be used with the **\v** statement in other topics to create links to this topic.
$	Specifies a topic title. Windows Help uses the topic title to identify the topic in the Search and History dialog boxes. The *text* parameter can be any combination of characters including spaces.
+	Specifies the browse-sequence identifier. Windows Help adds topics having an identifier to the browse sequence and allows users to view the topics by using the browse buttons. The *text* parameter can be a combination of letters and digits. Windows Help determines the order of topics in the browse sequence by sorting the identifier alphabetically. If two topics have the same identifier, Windows Help assumes that the topic that was compiled first is to be displayed first. Windows Help uses the browse sequence identifier only if the browse buttons have been enabled by using the **BrowseButtons** macro.
K	Specifies a keyword. Windows Help displays all keywords in the Help file in the Search dialog box and allows a user to choose a topic to view by choosing a keyword. The *text* parameter can be any combination of characters including spaces. If the first character is the letter *K,* it must be preceded with an extra space or a semicolon. More than one keyword can be given by separating the keywords with semicolons (;). A topic cannot contain keywords unless it also has a topic title.
!	Specifies a Help macro. Windows Help executes the macro when the topic is displayed. The *text* parameter can be any Help macro.

If *n* is any letter (other than *K*), the footnote specifies an alternative keyword. Windows applications can search for topics having alternative keywords by using the HELP_MULTIKEY command with the **WinHelp** function.

text	Specifies the build tag, context string, topic title, browse-sequence number, keyword, or macro associated with the footnote. This parameter depends on the footnote type as specified by the *n* parameter.

Comments

Repetition of the footnote character, *n,* in the syntax is deliberate.

A topic can have more than one build-tag, context-string, keyword, and help-macro statement, but must not have more than one topic-title or browse-sequence-number statement.

In print-based documents, the **\footnote** statement creates a footnote. The footnote is anchored to the character immediately preceding the **\footnote** statement.

The characters in a context string must be alphanumeric and can include underscore characters (_) and periods (.).

The browse sequence string consists of a major sequence string and a minor sequence string, delimited by a colon:

{+}{\footnote {+} *major:minor*}

This syntax specifies disjoint sets of ordered browse sequences. The major sequence string determines which browse sequence a topic belongs to, while the minor sequence string determines its position. Minor sequence strings are sorted alphabetically, not numerically; to use numbers, they should be preceded with zeros so that they are all the same length. All topics with browse sequence strings that omit the major sequence string are placed on the same browse sequence.

A topic cannot have more than one build tag footnote. If a topic has a build tag footnote, it must be the first thing in that topic. The title, browse sequence, and macro must be in the first paragraph. Context strings and keywords may appear anywhere; if placed in the middle of a topic, jumps to that context string or keyword will bring you to the middle of that topic.

Example

The following example defines a topic titled "Short Topic". The context string "topic1" can be used to create links to this topic. The keywords "example topic" and "short topic" appear in the Search dialog box and can be used to choose the topic for viewing:

```
${\footnote Short Topic}
#{\footnote topic1}
K{\footnote example topic;short topic}
This topic has a title, context string, and two keywords.
\par
\page
```

See Also

\v

\fs RTF statement

\fs*n*

The **\fs** statement sets the size of the font. The new font size applies to all subsequent text up to the next **\plain** or **\fs** statement.

Parameter	Description
n	Specifies the size of the font, in half points.

Comments

The **\fs** statement does not set the font face; use the **\f** statement instead.

No **\plain** or **\fs** statement is required if the **\fs** statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

If the **\fs** statement is not given, the default font size is 24.

Example

The following example sets the size of the font to 10 points:

```
{\fs20 This line is in 10 point type.}
\par
```

See Also

\plain
\f

\' RTF statement

\'*hh*

The **\'** statement converts the specified hexadecimal number into a character value and inserts the value into the Help file. The appearance of the character when displayed depends on the character set specified for the Help file.

Parameter	Description
hh	Specifies a two-digit hexadecimal value.

Comments

Since the Help Compiler does not accept character values greater than 127, the **\'** statement is the only way to insert such character values into the Help file.

Example

The following example inserts a trademark in a Help file that uses the **\ansi** statement to set the character set:

```
ABC\'99 is a trademark of the ABC Product Corporation.
```

See Also

\ansi
\mac
\pc

\i RTF statement

\i

The **\i** statement starts italic text. The statement applies to all subsequent text up to the next **\plain** or **\i0** statement.

Comments

No **\plain** or **\i0** statement is required if the **\i** statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

Example

The following example sets "not" to italic:

```
You must {\i not} save the file without first setting the
Auto option.
```

See Also

\b
\plain
\scaps

\intbl RTF statement (3.1)

\intbl

The **\intbl** statement marks subsequent paragraphs as part of a table. The statement applies to all subsequent paragraphs up to the next **\row** statement.

Comments

This statement was first supported in Help Compiler version 3.1.

Example

The following example creates a three-column table having two rows:

```
\cellx1440\cellx2880\cellx4320
\intbl
Row 1 Column 1\cell
Row 1 Column 2\cell
Row 1 Column 3\cell \row
\intbl
Row 2 Column 1\cell
Row 2 Column 2\cell
Row 2 Column 3\cell \row \pard
```

See Also

\cell
\cellx
\row
\trgaph
\trleft
\trowd

\keep RTF statement

\keep

The **\keep** statement prevents Windows Help from wrapping text to fit the Help window. The statement applies to all subsequent paragraphs up to the next **\pard** statement.

Comments

If the text in a paragraph exceeds the width of the Help window, Help displays a horizontal scroll bar.

In print-based documents, the **\keep** statement keeps paragraphs intact.

See Also

\keepn
\line

\keepn RTF statement

\keepn

The **\keepn** statement creates a non-scrolling region at the top of the Help window for the given topic. The **\keepn** statement applies to all subsequent paragraphs up to the next **\pard** statement. All paragraphs with this paragraph property are placed in the non-scrolling region.

Comments

If a **\keepn** statement is used in a topic, it must be applied to the first paragraph in the topic (and subsequent paragraphs as needed). The Help compiler displays an error message and does not create a non-scrolling region if paragraphs are given before the **\keepn** statement. Only one non-scrolling region per topic is allowed.

Windows Help formats, aligns, and wraps text in the non-scrolling region just as it does in the rest of the topic. It separates the non-scrolling region from the rest of the Help window with a horizontal bar. Windows Help sets the height of the non-scrolling region so that all all paragraphs in the region can be viewed if the help window is large enough. If the window is smaller than the non-scrolling region, the user will be unable to view the rest of the topic. For this reason, the non-scrolling region is typically reserved for a single line of text specifying the name or title of the topic.

In print-based documents, the **\keepn** statement keeps the subsequent paragraph with the paragraph that follows it.

See Also

\keep
\page

\li RTF statement

\li*n*

The **\li** statement sets the left indent for the paragraph. The indent applies to all subsequent paragraphs up to the next **\pard** or **\li** statement.

Parameter	Description
n	Specifies the indent, in twips. The value can be either positive or negative.

Comments

If the **\li** statement is not given, the left indent is zero by default. Windows Help automatically provides a small left margin so that if no indent is specified the text does not start immediately at the left edge of the Help window.

Specifying a negative left indent moves the starting point for a line of text to the left of the default left margin. If the negative indent is large enough, the start of the text may be clipped by the left edge of the help window.

Example

The following example uses the left indent and a tab stop to make a bulleted list. In this example, font number 0 is assumed to be the Symbol font:

```
Use the Auto command to:
\par
\tx360\li360\fi-360
{\f0\'B7}
\tab
Save files automatically
\par
{\f0\'B7}
\tab
Prevent overwriting existing files
\par
{\f0\'B7}
\tab
Create automatic backup files
\par \pard
```

See Also

\fi
\pard
\ri

\line RTF statement

\line

The **\line** statement breaks the current line without ending the paragraph. Subsequent text starts on the next line and is aligned and indented according to the current paragraph properties.

See Also

\par

\mac RTF statement

\mac

The **\mac** statement sets the Apple Macintosh character set.

See Also

\ansi
\pc

\page RTF statement

\page

The **\page** statement marks the end of a topic.

Comments

In a print-based document, the **\page** statement creates a page break.

Example

The following example shows a complete topic:

```
${\footnote Short Topic}
#{\footnote short_topic}
Most topics in a topic file consist of topic-title and
context-string statements followed by the topic text. Every
topic ends with a {\b \\page} statement.
\par
\page
```

See Also

\par

\par RTF statement

\par

The **\par** statement marks the end of a paragraph. The statement ends the current line of text and moves the current position to the left margin and down by the current line-spacing and space-after-paragraph values.

Comments

The first line of text after a **\par**, **\page**, or **\sect** statement marks the start of a paragraph. When a paragraph starts, the current position is moved down by the current space-before-paragraph value. Subsequent text is formatted using the current text alignment, line spacing, and left, right, and first-line indents.

Example

The following example has three paragraphs:

```
\ql
This paragraph is left-aligned.
\par \pard
\qc
This paragraph is centered.
\par \pard
\qr
This paragraph is right-aligned.
\par
```

See Also

\line
\pard
\sect

\pard RTF statement

\pard

The **\pard** statement restores all paragraph properties to default values.

Comments

If the **\pard** statement appears anywhere before the end of a paragraph (that is, before the **\par** statement), the default properties apply to the entire paragraph.

The default paragraph properties are as follows:

Property	Default
Alignment	Left-aligned
First-line indent	0
Left indent	0
Right indent	0
Space before	0
Space after	0
Line spacing	Tallest character
Tab stops	None
Borders	None
Border style	Single-width

See Also

\par

\pc RTF statement

\pc

The **\pc** statement sets the OEM character set (also known as code page 437).

See Also

\ansi
\mac

\pich RTF statement

\pich*n*

The **\pich** statement specifies the height of the picture. This statement must be used in conjunction with a **\pict** statement.

Parameter	Description
n	Specifies the height of the picture, in twips or pixels, depending on the picture type. If the picture is a metafile, the width is in twips; otherwise, the width is, in pixels.

See Also

\pict
\picw

\pichgoal RTF statement

\pichgoal*n*

The **\pichgoal** statement specifies the desired height of a picture. If necessary, Windows Help stretches or compresses the picture to match the requested height. This statement must be used in conjunction with a **\pict** statement.

Parameter	Description
n	Specifies the desired height, in twips.

Comments

The **\pichgoal** statement is not supported for metafiles. Applications should use the **\pich** statement, instead.

See Also

\pich
\pict
\picwgoal

\picscalex RTF statement

\picscalex*n*

The **\picscalex** statement specifies the horizontal scaling value. This statement must be used in conjunction with a **\pict** statement.

Parameter	Description
n	Specifies the scaling value as a percentage. If this value is greater than 100, the bitmap or metafile is enlarged.

Comments

If the **\picscalex** statement is not given, the default scaling value is 100.

See Also

\pict
\picscaley

\picscaley RTF statement

\picscaley*n*

The **\picscaley** statement specifies the vertical scaling value. This statement must be used in conjunction with a **\pict** statement.

Parameter	Description
n	Specifies the scaling value as a percentage. If this value is greater than 100, the bitmap or metafile is enlarged.

Comments

If the **\picscaley** statement is not given, the default scaling value is 100.

See Also

\pict
\picscalex

\pict RTF statement

\pict*picture-statementspicture-data*

The **\pict** statement creates a picture. A picture consists of hexadecimal or binary data representing a bitmap or metafile.

Parameter	Description
picture-statements	Specifies one or more statements defining the type of picture, the dimensions of the picture, and the format of the picture data. It can be a combination of the following statements:

Statement	Descripton
\wbitmap	Specifies a Windows bitmap.
\wmetafile	Specifies a Windows metafile.
\picw	Specifies the picture width.
\pich	Specifies the picture height.
\picwgoal	Specifies the desired picture width.
\pichgoal	Specifies the desired picture height.
\picscalex	Specifies the horizontal scaling value.
\picscaley	Specifies the vertical scaling value.
\wbmbitspixel	Specifies the number of bits per pixel.
\wbmplanes	Specifies the number of planes.
\wbmwidthbytes	Specifies the bitmap width, in bytes.
\bin	Specifies binary picture data.

picture-data	Specifies hexadecimal or binary data representing the picture. The picture data follows the last picture statement.

Comments

If a data format is not specified, the default format is hexadecimal.

See Also

\bin
\pich
\pichgoal
\picscalex
\picscaley
\picw
\picwgoal
\wbitmap
\wbmbitspixel
\wbmplanes
\wbmwidthbytes
\wmetafile

\picw RTF statement

\picw*n*

The **\picw** statement specifies the width of the picture. This statement must be used in conjunction with a **\pict** statement.

Parameter	Description
n	Specifies the width of the picture, in twips or pixels, depending on the picture type. If the picture is a metafile, the width is in twips; otherwise, the width is in pixels.

See Also

\pict
\pich

\picwgoal RTF statement

\picwgoal*n*

The **\picwgoal** statement specifies the desired width of the picture, in twips. If necessary, Windows Help stretches or compresses the picture to match the requested height. This statement must be used in conjunction with a **\pict** statement.

Parameter	Description
n	Specifies the desired width, in twips.

Comments

The **\picwgoal** statement is not supported for metafiles. Applications should use the **\picw** statement, instead.

See Also

\pict
\picw
\pichgoal

\plain RTF statement

\plain

The **\plain** statement restores the character properties to default values.

Comments

The default character properties are as follows:

Property	Default
Bold	Off
Italic	Off
Small caps	Off
Font	0
Font size	24

See Also

\b, **\i**
\scaps
\f
\fs

\qc RTF statement

\qc

The **\qc** statement centers text between the current left and right indents. The statement applies to subsequent paragraphs up to the next **\pard** statement or text-alignment statement.

Comments

If a **\ql**, **\qr**, or **\qc** statement is not given, the text is left-aligned by default.

See Also

\pard
\ql
\qr

\ql RTF statement

\ql

The **\ql** statement aligns text along the left indent. The statement applies to subsequent paragraphs up to the next **\pard** statement or text-alignment statement.

Comments

If a **\ql**, **\qr**, or **\qc** statement is not given, the text is left-aligned by default.

See Also

\pard
\qc
\qr

\qr RTF statement

\qr

The **\qr** statement aligns text along the right indent. The statement applies to subsequent paragraphs up to the next **\pard** statement or text-alignment statement.

Comments

If a **\ql**, **\qr**, or **\qc** statement is not given, the text is left-aligned by default.

See Also

\pard
\qc
\ql

\ri RTF statement

\ri*n*

The **\ri** statement sets the right indent for the paragraph. The indent applies to all subsequent paragraphs up to the next **\pard** or **\ri** statement.

Parameter	Description
n	Specifies the right indent, in twips. It can be a positive or negative value.

Comments

If the **\ri** statement is not given, the right indent is zero by default. Windows Help automatically provides a small right margin so that when no right indent is specified, the text does not end abruptly at the right edge of the Help window.

Windows Help never displays less than one word for each line in a paragraph even if the right indent is greater than the width of the window.

Example

In the following example, the right and left indents are set to one inch and the subsequent text is centered between the indents:

```
\li1440\ri1440\qc
Windows Help\line
Sample File\line
```

See Also

\li
\pard

\row RTF statement

\row

The **\row** statement marks the end of a table row. The statement ends the current row and begins a new row by moving down pass the end of the longest cell in the row. The next **\cell** statement specifies the text of the leftmost cell in the next row.

Comments
This statement was first supported in the Help Compiler version 3.1.

Example
The following example creates a table having four rows and two columns:

```
\cellx2880\cellx5760
\intbl
Row 1, Column 1\cell
Row 1, Column 2\cell \row
\intbl
Row 2, Column 1\cell
Row 2, Column 2\cell \row
\intbl
Row 3, Column 1\cell
Row 3, Column 2\cell \row
\intbl
Row 4, Column 1\cell
Row 4, Column 2\cell \row
\par \pard
```

See Also

\cell
\cellx
\intbl

\rtf RTF statement

\rtf*n*

The **\rtf** statement identifies the file as a rich-text format (RTF) file and specifies the version of the RTF standard used.

Parameter	Description
n	Specifies the version of the RTF standard used. For the Help Compiler version 3.1, this parameter must be 1.

Comments

The **\rtf** statement must follow the first open brace in the Help file. A statement specifying the character set for the file must also follow the **\rtf** statement.

See Also

\ansi

\sa RTF statement

\sa*n*

The **\sa** statement sets the amount of vertical spacing after a paragraph. The vertical space applies to all subsequent paragraphs up to the next **\pard** or **\sa** statement.

Parameter	Description
n	Specifies the amount of vertical spacing, in twips.

Comments

If the **\sa** statement is not given, the vertical spacing after a paragraph is zero by default.

See Also

\sb
\pard

\sb RTF statement

\sb*n*

The **\sb** statement sets the amount of vertical spacing before the paragraph. The vertical space applies to all subsequent paragraphs up to the next **\pard** statement or **\sb** statement.

Parameter	Description
n	Specifies the amount of vertical spacing, in twips.

Comments

If the **\sb** statement is not given, the vertical spacing before the paragraph is zero by default.

See Also

\sa
\pard

\scaps RTF statement

\scaps

The **\scaps** statement starts small-capital text. The statement converts all subsequent lowercase letters to uppercase before displaying the text. This statement applies to all subsequent text up to the next **\plain** or **\scaps0** statement.

Comments

The **\scaps** statement does not affect uppercase letters.

No **\plain** or **\scaps0** statement is required if the **\scaps** statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

The **\scaps** statement does not reduce the point size of the text. To reduce point size, the **\fs** statement must be used.

Example

The following example displays the key name ENTER in small capitals:

```
Press the {\scaps enter} key to complete the action.
```

See Also

\fs
\plain

\sect RTF statement

\sect

The **\sect** statement marks the end of a section and paragraph.

See Also

\par

\sl RTF statement

\sl*n*

The **\sl** statement sets the amount of vertical space between lines in a paragraph. The vertical space applies to all subsequent paragraphs up to the next **\pard** or **\sl** statement.

Parameter	Description
n	Specifies the amount of vertical spacing, in twips. If this parameter is a positive value, Windows Help uses this value if it is greater than the tallest character. Otherwise, Windows Help uses the height of the tallest character as the line spacing. If this parameter is a negative value, Windows Help uses the absolute value of the number even if the tallest character is taller.

Comments

If the **\sl** statement is not given, Windows Help automatically sets the line spacing by using the tallest character in the line.

See Also

\pard

\strike RTF statement

\strike

The **\strike** statement creates a hot spot. The statement is used in conjunction with a **\v** statement to create a link to another topic. When the user chooses a hot spot, Windows Help displays the associated topic in the Help window.

The **\strike** statement applies to all subsequent text up to the next **\plain** or **\strike0** statement.

Comments

No **\plain** or **\strike0** statement is required if the **\strike** statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

The **\strike** statement creates the same type of hot spot as the **\uldb** statement.

In print-based documents, or whenever it is not followed by **\v**, the **\strike** statement creates strikeout text.

Example

The following example creates a hot spot for a topic. When displayed, the hot-spot text, "Hot Spot," is green and has a solid line under it:

```
{\strike Hot Spot}{\v Topic}
```

See Also

\ul
\uldb
\v

\tab RTF statement

\tab

The **\tab** statement inserts a tab character (ASCII character code 9).

Comments

The tab character (ASCII character code 9) has the same effect as the **\tab** statement.

See Also

\tqc
\tqr
\tx

\tqc RTF statement

\tqc

The **\tqc** statement is used with the **\tx** statement to create a tab stop where text is centered. For example, the following statement creates a centered tab stop at 2880 twips:

```
\tqc\tx2880
```

See Also

\tab
\tqr
\tx

\tqr RTF statement

\tqr

The **\tqr** statement is used with the **\tx** statement to create a tab stop where text right-justified. For example, the following statement creates a right-justified tab stop at 2880 twips:

```
\tqr\tx2880
```

See Also

\tab
\tqc
\tx

\trgaph RTF statement (3.1)

\trgaph*n*

The **\trgaph** statement specifies the amount of space between text in adjacent cells in a table. For each cell in the table, Windows Help uses the space to calculate the cell's left and right margins. It then uses the margins to align and wrap the text in the cell. Windows Help applies the same margin widths to each cell ensuring that paragraphs in adjacent cells have the specified space between them.

The **\trgaph** statement applies to cells in all subsequent rows of a table up to the next **\trowd** statement.

Parameter	Description
n	Specifies the space, in twips, between text in adjacent cells. If this parameter exceeds the actual width of the cell, the left and right margins are assumed to be at the same position in the cell.

Comments

The width of the left margin in the first cell is always equal to the space specified by this statement. The **\trleft** statement is typically used to move the left margin to a position similar to the left margins in all other cells.

This statement was first supported in the Help Compiler version 3.1.

Example

The following example creates a three-column table with one-quarter inch space between the text in the columns:

```
\trgaph360 \cellx1440\cellx2880\cellx4320
\intbl
Row 1 Column 1\cell
Row 1 Column 2\cell
Row 1 Column 3\cell \row
\intbl
Row 2 Column 1\cell
Row 2 Column 2\cell
Row 2 Column 3\cell \row \pard
```

See Also

\cell
\cellx
\intbl
\row
\trleft
\trowd

\trleft RTF statement

\trleft*n*

The **\trleft** statement sets the position of the left margin for the first (leftmost) cell in a row of a table. This statement applies to the first cell in all subsequent rows of the table up to the next **\trowd** statement.

Parameter	Description
n	Specifies the relative position, in twips, of the left margin. This parameter can be a positive or negative number. The final position of the left margin is the sum of the current position and this value.

Comments

This statement was first supported in the Help Compiler version 3.1.

Example

The following example creates a three-column table with one-quarter inch space between the text in the columns. The left margin in the first cell is flush with the left margin of the Help window:

```
\trgaph360\trleft-360 \cellx1440\cellx2880\cellx4320
\intbl
Row 1 Column 1\cell
Row 1 Column 2\cell
Row 1 Column 3\cell \row
\intbl
Row 2 Column 1\cell
Row 2 Column 2\cell
Row 2 Column 3\cell \row \pard
```

See Also

\cell
\cellx
\intbl
\row
\trgaph
\trowd

\trowd RTF statement

\trowd

The **\trowd** statement sets default margins and cell positions for subsequent rows in a table.

Comments

This statement was first supported in the Help Compiler version 3.1.

See Also

\cell
\cellx
\intbl
\row
\trgaph
\trleft

\trqc RTF statement

\trqc

The **\trqc** statement directs Windows Help to dynamically adjust the width of table columns to fit in the current window.

Comments

In a print-based document, the **\trqc** statement centers a table row with respect to its containing column.

Windows Help will not resize a table to smaller than the widths specified in the **\trqc** statement. Therefore, the table should be created in the smallest size in which it would ever be displayed. All columns in the table are sized proportionally.

This statement was first supported in the Help Compiler version 3.1.

See Also

\trowd
\trql

\trql RTF statement

\trql

The **\trql** statement aligns the text in each cell of a table row to the left.

Comments
This statement was first supported in the Help Compiler version 3.1.

See Also

\trowd
\trqc

\tx RTF statement

\tx*n*

The **\tx** statement sets the position of a tab stop. The position is relative to the left margin of the Help window. A tab stop applies to all subsequent paragraphs up the next **\pard** statement.

Parameter	Description
n	Specifies the tab stop position, in twips.

Comments

If the **\tx** statement is not given, tab stops are set at every one-half inch by default.

See Also

\tab
\tqc
\tqr

\ul RTF statement

\ul

The **\ul** statement creates a link to a pop-up topic. The statement is used in conjunction with a **\v** statement to create a link to another topic. When the user chooses the link, Windows Help displays the associated topic in a pop-up window.

The **\ul** statement applies to all subsequent text up to the next **\plain** or **\ul0** statement.

Comments

No **\plain** or **\ul0** statement is required if the **\ul** statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

In print-base documents, or whenever it is not followed by **\v**, the **\ul** statement creates a continuous underline.

Example

The following example creates a pop-up link for a topic. When displayed, the link text, "Popup Link," is green and has a dotted line under it:

```
{\ul Popup Link}{\v PopupTopic}
```

See Also

\strike
\uldb
\v

\uldb RTF statement

\uldb

The **\uldb** statement creates a hot spot. This statement is used in conjunction with a **\v** statement to create a link to another topic. When the user chooses a hot spot, Windows Help displays the associated topic in the Help window.

The **\uldb** statement applies to all subsequent text up to the next **\plain** or **\uldb0** statement.

Comments

No **\plain** or **\uldb0** statement is required if the **\uldb** statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

The **\uldb** statement creates the same type of hot spot as the **\strike** statement.

Example

The following example creates a hot spot for a topic. When displayed, the hot-spot text, "Hot Spot," is green and has a solid line under it:

```
{\uldb Hot Spot}{\v Topic}
```

See Also

\strike
\ul
\v

\v RTF statement

{**\v** *context-string*}

The **\v** statement creates a link to the topic having the specified context string. The **\v** statement is used in conjunction with the **\strike**, **\ul**, and **\uldb** statements to create hot spots and links to topics.

Parameter	Description
context-string	Specifies the context string of a topic in the Help file. The string can be any combination of characters, except spaces, and must also be specified in a context-string **\footnote** statement in some topic in the Help file.

Comments

If the context string is preceded by a percent sign (%), Windows Help displays the associated hot spot or link without applying the standard underline and color. If the context string is preceded by an asterisk (*), Windows Help displays the associated hot spot or link with an underline but without applying the standard color.

In print-based documents, the **\v** statement creates hidden text.

For links or hot spots, the syntax of the **\v** statement is as follows:

[**%**|*****] *context* [**>***secondary-window*] [**@***filename*]

In this syntax, *secondary-window* is the name of the secondary window to jump to. When the secondary window is not specified, the jump is to the same window as the current help topic is using. To jump to the main window, specify "main" for this parameter. This parameter may not be used with pop-up windows.

The *filename* parameter specifies a jump to a topic in a different help file.

For a macro hotspot, the syntax of the **\v** statement is as follows:

[**%**|*****] **!** *macro* [**;***macro*][**;**...]

Example

The following example creates a hot spot for the topic having the context string "Topic". Windows Help applies an underline and the color green the text "Hot Spot" when it displays the topic:

```
{\uldb Hot Spot}{\v Topic}
```

See Also

\footnote
\strike
\ul
\uldb

\wbitmap RTF statement

\wbitmap*n*

The **\wbitmap** statement sets the picture type to Windows bitmap. This statement must be used in conjunction with a **\pict** statement.

Parameter	Description
n	Specifies the bitmap type. This parameter is zero for a logical bitmap.

Comments

The **\wbitmap** statement is optional; if a **\wmetafile** statement is not specified, the picture is assumed to be a Windows bitmap.

Example

The following example creates a 32-by-8 pixel monochrome bitmap:

```
{
\pict \wbitmap0\wbmbitspixel1\wbmplanes1\wbmwidthbytes4\picw32\pich8
3FFFFFFC
F3FFFFCF
FF3FFCFF
FFF3CFFF
FFFC3FFF
FFCFF3FF
FCFFFF3F
CFFFFFF3
}
```

See Also

bmc
bml
bmr
\pict
\wmetafile

\wbmbitspixel RTF statement

\wbmbitspixel*n*

The **\wbmbitspixel** statement specifies the number of consecutive bits in the bitmap data that represent a single pixel. This statement must be used in conjunction with the **\pict** statement.

Parameter	Description
n	Specifies the number of bits per pixel.

Comments

If the **\wbmbitspixel** statement is not given, the default bits per pixel value is 1.

See Also

\pict
\wbitmap
\wbmplanes

\wbmplanes RTF statement

\wbmplanes*n*

The **\wbmplanes** statement specifies the number of color planes in the bitmap data. This statement must be used in conjunction with a **\pict** statement.

Parameter	Description
n	Specifies the number of bitmap planes.

Comments

If the **\wbmplanes** statement is not given, the default number of planes is 1.

See Also

\pict
\wbitmap
\wbmbitspixel

\wbmwidthbytes RTF statement

\wbmwidthbytes*n*

The **\wbmwidthbytes** statement specifies the number of bytes in each scan line of the bitmap data. This statement must be used in conjunction with the **\pict** statement.

Parameter	Description
n	Specifies the width of the bitmap, in bytes.

See Also

\pict
\wbitmap

\wmetafile RTF statement

\wmetafile*n*

The **\wmetafile** statement sets the picture type to a Windows metafile. This statement must be used in conjunction with the **\pict** statement.

Parameter	Description
n	Specifies the metafile type. This parameter must be 8.

Comments

Windows Help expects the hexadecimal data associated with the picture to represent a valid Windows metafile. By default, Windows Help sets the MM_ANISOTROPIC mapping mode prior to displaying the metafile. To ensure that the picture is displayed correctly, the metafile data must either set the window origin and extents by using the **SetWindowOrg** and **SetWindowExt** records or set another mapping mode by using the **SetMapMode** record.

Example

The following example creates a picture using a metafile:

```
{{\pict\wmetafile8\picw2880\pich2880
0100090000034f0000000200090000000000
050000000b0200000000050000000c026400
6400090000001d066200ff00640064000000
000008000000fa020000020000000000000000
040000002d0100000500000014020000000
0500000013026400640005000000014020000
64000500000013026400000008000000fa02
000000000000000000000040000002d010100
04000000f00100000300000000004e0dff00
870020000050000020000000000000000000}
\par }
```

See Also

bmc
bml
bmr
\pict
\wbitmap