BackMenu - The Pop-Up Menu

BackMenu is a program that allows you to define a pop-up menu on the Windows 3.1 background and use it to quickly run applications and tools. The idea came from using X-Windows and Sunview, where the user has a root menu that can be configured.

It allows you to describe a set of actions and associate command lines with them. To bring up the menu simply click with the mouse button (left, middle or right) on the Windows desktop. This will bring up the description list. Selecting a description will cause the appropriate command line to be executed.

Help for BackMenu is available on:-

- What is the format for a BackMenu menu file?
- How do I create invisible sub-menus?
- How do I actually create a menu?
- How do I make BackMenu start applications automatically?
- How do I make BackMenu the Windows shell?
- How can I use BackMenu from another application?
- What keyword commands are there?
- How do I construct a command line?
- What is an alias or tool?
- How do I write an external keyword for BackMenu?

BackMenu - Creating a menu.

A BackMenu menu is simply a set of lines (one for each description/command line pair). Additionally, comments may be placed within the menu file. These are lines that start with a semicolon (;). Any text following it on that line is ignored.

The syntax for a menu item is given below. Items contained within square brackets are optional. For more detailed information on the command line parameters, see the BackMenu command syntax. For more details on the keywords, see the keyword summary.

```
Menu Text,[@]command-line
```

or

```
Menu Text, $Keyword [Keyword Parameters]
```

Blank lines may be placed between items, to cause a separator line to be placed in the menu.

BackMenu also supports cascading menus.   A cascading menu item is one that has multiple options associated with it. By clicking on the item, another menu appears with the options on it. It is even possible to have further cascading menus associated with one or more items within that menu.

Associating multiple items with a single menu item is simple. The definition is:-

```
>Item Name
   Sub menu item 1, command-line
   Sub menu item 2, command-line
   ...
!
```

That is, the sub-menu is enclosed between a '>' and a '!'. The text that follows the '>' is the text to appear in the menu (a description of the sub-menu). For example :-

```
;
; Example BackMenu menu file
;
Word-processing, {DIR=c:\work}Write
Spreadsheet, C:\Windows\Excel\Excel

Program Manager Groups, $Groups
>BackMenu Options
   About...,$About

   Set Options, $SetOptions
   Edit Menu, $EditMenu

   Exit Windows, $ExitWindows NoConfirm
!
```

Defining items within a sub-menu is done in exactly the same way as for a main menu (even to the extent of adding a sub-menu to this sub-menu). E.g.:-

```
>Editors
   Andy's editor, AE.EXE
   Notepad, NOTEPAD.EXE
```

```
      >Word processors
         Word for Windows, WORD.EXE
         Windows Write, WRITE.EXE
      !
      Multi-Pad, MULTIPAD.EXE
   !
   ...
```

There is no limit to the number of items you can have in any particular menu. BackMenu will split the menu into multiple columns when it gets over a certain size. How big is that size? See the $SetOptions dialog box.

BackMenu - Invisible Sub-Menus

You can have sub-menus which are processed by BackMenu but do not form part of the menu displayed. To do this, use '<' rather than '>' when defining the sub-menu. How is this useful? If you use the auto run (@) operator, you can create a little start-up menu which contains a list of everything you want to run, but which won't appear in the menu. For example :-

```
;
; Some Auto run items
;
<Startup
   Clock,@clock.exe
   File Manager,@{XPOS=50 YPOS=100}File Manager
!
>Main
   Command Window,dosprmpt.pif
...
```

Note that anything contained between '<' and '!' will **not** be visible. If you define any sub menus or other commands in a hidden menu, they won't appear, regardless of whether they are visible or not.

BackMenu - Editing and Checking the Menu

You can use any text editor to create the BackMenu menu file. By default, BackMenu looks for a file called BACKMENU.INI in the same directory as BACKMENU.EXE. The menu file name can be changed by using the $SetOptions dialog box.

BackMenu checks the menu file before attempting to create your menu. If BackMenu encounters an error, you have the option to load an editor and edit the menu file, and the currently loaded menu will be left unchanged. BackMenu will insert a comment below the line which caused the error (or at the end of the file if the error is a missing "!"). The editor used is configurable and can be set from the $SetOptions dialog box. If no editor is specified (i.e. the entry is blank), BackMenu will default to using Notepad. Currently, BackMenu checks for lines that are too long, contain invalid keywords or problems in the definition of cascading menus.

BackMenu will automatically monitor the menu file and reload it if the file is modified. It checks the file every time the menu is accessed and a small delay is encountered if the menu has to be reloaded (along with a message and the usual animated cursor). The menu is parsed at this time as well and so it is possible for an error dialog box to appear as described above.

BackMenu - Auto-running Applications

If you install BackMenu as the shell, it will start any programs contained within the Program Manager start-up group. It is also possible, however, to mark applications within the menu so that, when BackMenu is invoked, the marked applications are run automatically. To do this, insert an at symbol (@) before the command line for that item. E.g.

```
;
; Example of an menu using Auto-Run feature
;
>My Apps
   Clock, @clock.exe
   Screen Saver, @c:\winapps\saver\Saver.exe

   Free Memory, FREEMEM.EXE
!
```

In the example above, clock.exe and saver.exe will be executed when BackMenu is run. If you use the choose to reload the menu file, however, the @ will be ignored. This allows you to use BackMenu to decide which program you want running.

There may be times when you've selected some applications to run, but don't actually want them to start up when you run BackMenu. You can do this by holding down **either shift key** while BackMenu is loading. This will cause BackMenu to ignore any @'s that you've placed in the menu file, and also ignore the start-up group (if BackMenu is the shell).

BackMenu - Using it as the Start-Up Shell

BackMenu can be used in place of Program Manager as the default shell for Windows. To install it simply edit SYSTEM.INI which is in the windows directory. In the [boot] section of the file there is a line which will read:-

```
    shell=progman.exe
```

Replace it with:-

```
    shell=backmenu.exe
```

and you're away. We've noticed that some ill-behaved installation routines set this back without asking; sorry, we can't alter other people's inconsiderate coding! Similarly, some install routines expect Program Manager to be running; just start a copy before starting the installation.

If BackMenu and its DLLs are not on the DOS path, BackMenu will not be usable as the shell because of the way Windows finds DLLs. To get round this problem, the program BDSHELL.EXE is provided, which makes the conditions right for loading BackMenu and then loads BackMenu. If you know BackMenu is not on the path or experience errors when starting Windows with BackMenu set as the shell, use the line:-

```
    shell=c:\winapps\backdesk\bdshell.exe
```

in SYSTEM.INI. Replace c:\winapps\backdesk with the actual location of the file. BDSHELL must be in the same directory as BackMenu and its associated files.

BackMenu - Access From Other Applications

The BackDesk DLL (BACKDESK.DLL) contains three accessible functions that can be called from other applications to let you use some of the features of BackMenu. They are:-

integer **BD_IsBackMenuRunning()**

Is BackMenu installed and running? This function returns an integer which is 1 if BackMenu is running and 0 if not.

ActivateBackMenu()

Allows the menu to be popped up by another application. This function takes no parameters.

integer **BM_Execute(***integer*,*string*,*integer***)**

Execute a application using the BackMenu command line syntax. This function takes three parameters. The first is an integer which contains the window identifier for the application making the call. For macro languages, use 0 for this value. The second parameter is the BackMenu command string. Please note that you can not execute any keywords using this function. The third parameter describes how the resulting application's window is to appear. The values it can take are as follows:-

0	Hidden
1	Normal
2	Minimised
3	Maximised

Any attributes in the BackMenu command will override the action of this value.

Here are a couple of Microsoft Word-For-Windows macro that demonstrate how to use these functions

```
Declare Sub ActivateBackMenu Lib "BACKDESK.DLL"
Declare Function BD_IsBackMenuRunning Lib "BACKDESK.DLL" As Integer

Sub MAIN
   If BD_IsBackMenuRunning <> 0 Then
      ActivateBackMenu
   Else
      MsgBox "BackMenu isn't running"
   End If
End Sub



Declare Function BM_Execute(Owner As Integer, Command$ As String, ShowAs
As Integer) Lib "BACKDESK.DLL" As Integer

Sub MAIN
   ret = BM_Execute(0, "{XPOS=100 YPOS=100}NOTEPAD.EXE", 1)
End Sub
```

Writing similar Excel macros is left as an exercise for the reader.

Keyword Commands

Click on the command you want help on. In the full descriptions, optional parameter are enclosed in square brackets [] and replaceable parameters are enclosed in angle brackets <>.

From V3.10, it is possible to extend the keyword commands available by writing external keywords and installing them into BackMenu. Some of the existing keywords have been moved over to the external keyword system and it is hoped, in time, that all keywords in BackMenu will be provided this way. For more information, see Writing External keyword DLLs for BackDesk V3.10

Keyword	**Short Description**
$About	Display version information
$AliasOptions	Change BackMenu alias settings
$AliasToBack	Send the aliases/tools behind all other windows
$AliasToFront	Bring the aliases/tools above all other windows
$ArrangeAlias	Arrange the aliases along one side of the screen
$CallDLL	Execute a function within another DLL
$DefaultPrinter	List the printers available and let you choose the default
$EditMenu	Edit the current menu.
$Execute	Choose an application to execute
$ExitWindows	Exit Windows
$Groups	Display a menu containing the Program Manager Groups
$HideAlias	Remove all aliases/tools from the screen.
$Info	Display system information
$LoadMenu	Load in another menu-file to replace the current.
$Net...	A set of network keywords that allow you to add/browse/delete network resources.
$NewAlias	Create a new alias
$ReloadMenu	Re-load in the default menu-file
$RemoveMenu	End the BackMenu application
$RestartWindows	Restart Windows
$RunFile	Execute a set of commands contained within a file
$SaveAlias	Save all the alias/tool positions.
$SetOptions	Set various BackMenu options
$ShowAlias	Bring all aliases/tools back onto the screen
$SnapAlias	Position aliases using a grid.
$StartSaver	Start the Windows screen saver
$Tasks	Display a menu containing all the executing applications.
$XKeywordManager	Provide management of external keywords

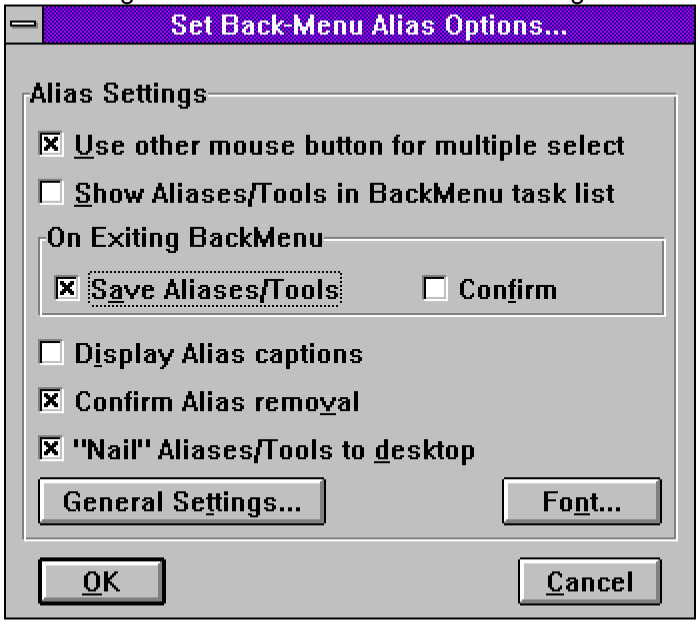
$About in BMKEYW.DLL

This one is simple. It presents you with a dialog box telling you all sorts of useful information, like the version number of the application and how to register.

If you've registered (thank you!) then you get to see you name as a reminder of your support for this product.

$AliasOptions

Want to fiddle with the way in which aliases (and tools) operate in BackMenu? Then look no further than the dialog box shown below. Click on something... Go on...



Set Back-Menu Alias Options...

Alias Settings

- [✓] Use other mouse button for multiple select
- [ ] Show Aliases/Tools in BackMenu task list

On Exiting BackMenu

- [✓] Save Aliases/Tools
- [ ] Confirm

- [ ] Display Alias captions
- [✓] Confirm Alias removal
- [✓] "Nail" Aliases/Tools to desktop

General Settings... | Font... | OK | Cancel

See Also

Other Alias Keywords

$AliasToFront $AliasToBack $HideAlias $ShowAlias $ArrangeAlias $SnapAlias $NewAlias $SaveAlias

Other ways of setting things

$SetOptions BackDrop Control Panel

Use other mouse button for multiple select

You can select a whole group of aliases/tools by clicking on the desktop and dragging with the other mouse button (i.e. the one not being used to pop-up BackMenu). This might get in the way of other desktop applications (like Norton Desktop for Windows), so you can selectively enable/disable it.

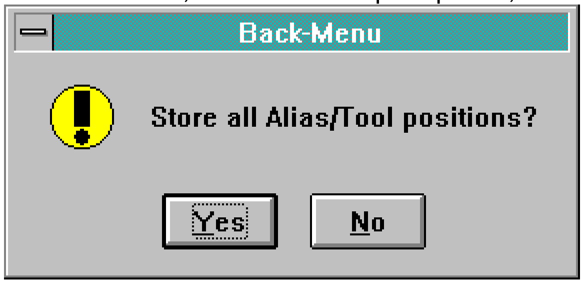
Show Alias/Tools in BackMenu task list

Choose whether you want all the tools and tasks to be displayed in the BackMenu task list (as provided by $Tasks). Aliases are prefixed with "*Alias* -" and tools with "*Tool* -". If you've got a lot of aliases, we suggest you un-check this option.

Note This does **not** remove the aliases and tools from the Task Manager task list.

Confirm save of Alias/Tool positions on exit

Normally when you exit Windows, or remove BackMenu, the positions of the various tools and aliases are saved automatically so they can be restored later. However, you can choose whether this happens. You can also set whether or not BackMenu will ask before saving by checking the *Confirm* option. If *Confirm* is set, BackMenu will prompt first, like this :-



Back-Menu

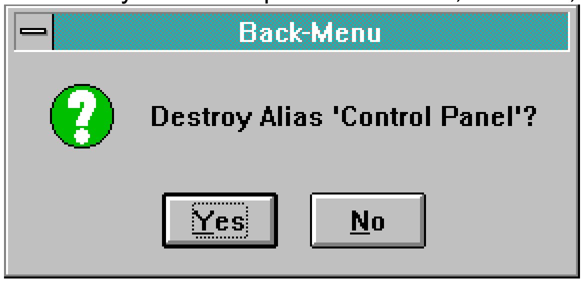
Store all Alias/Tool positions?

Display Alias Captions

Turns the title shown below an alias or tool on and off. Turning off alias captions means you can pack your aliases closer together, but you only then have the icon as an indication of what the alias does.

Confirm Alias Removal

If you highlight an alias and press "**CTRL+ALT+X**", or choose "**Close...**" from it's system menu, the alias is destroyed. If this option is checked, however, the alias will ask for conformation thus :-


Back-Menu

Destroy Alias 'Control Panel'?

General Settings

Clicking on this button brings up the BackDrop control panel which then lets you fiddle even more!

Font

Choose which font is used to display the text in the captions below the aliases.

Nail Alias/Tools to Desktop

Controls whether or not all aliases or tool can be moved from their current positions. This switch is very useful if you want to stop your beautiful aliases/tools layout from being spoilt by a twitch of the mouse. This setting is global, but can be overriden for an individual alias (so you can fine tune your layout to perfection). For more information see BackDrop Aliases and Tools - Configuring.

$AliasToBack [Selected]

Move aliases/tools behind all the other windows.

See Also

$AliasToFront $HideAlias $ShowAlias $ArrangeAlias $SnapAlias $NewAlias $SaveAlias $AliasOptions

Normally the keyword will operate on all of the aliases and tools. If **Selected** is specified, only those aliases and tools that gave been grouped will be modified.

Note to me: link to general stuff

$AliasToFront

Bring all of the aliases/tools to the front of all other windows.

See Also

$AliasToBack $HideAlias $ShowAlias $ArrangeAlias $SnapAlias $NewAlias $SaveAlias $AliasOptions

$ArrangeAlias [Selected]

Arrange the aliases along one side of the desktop. depending on the gravity set using the BackDrop control panel.

See Also

$AliasToFront $AliasToBack $HideAlias $ShowAlias $SnapAlias $NewAlias $SaveAlias $AliasOptions

$CallDLL <DLL Name> <Function Name> <String>

The **$CallDLL** is a powerful hook, through which the functionality of BackMenu can be extended. Programmers may write functions within a DLL that performs some function, and have it executed through BackMenu. Parameters may be passed to the function in the form of a string as this allows the most flexible way to pass parameters of different type. The % and * characters may be used to prompt for a parameter or get a file name from a list as with the other keywords. A prototype for a C function that could be called from BackMenu is:-

```
/*-----------------------------------------------------*/
void  FAR PASCAL  ShowBox(LPSTR Params)

{
     MessageBox(GetActiveWindow(),Params,"BackMenu DLL Test",
             MB_OK);
}
/*-----------------------------------------------------*/
```

A Turbo Pascal program that would work is as follows:

```
LIBRARY CommandLine;

{Sample procedure callable using $CALLDLL from BackMenu.}
{Not even slightly copyright.}

USES
   Strings, WinProcs, WinTypes;

PROCEDURE ShowBox(Params : pChar); EXPORT;
BEGIN

     MessageBox(GetActiveWindow, Params, 'BackMenu DLL Test', mb_Ok)

END; {PROCEDURE CommandLine}

EXPORTS
   ShowBox ;

BEGIN

END.
```

If you write any useful DLLs which use this facility, please let us know.

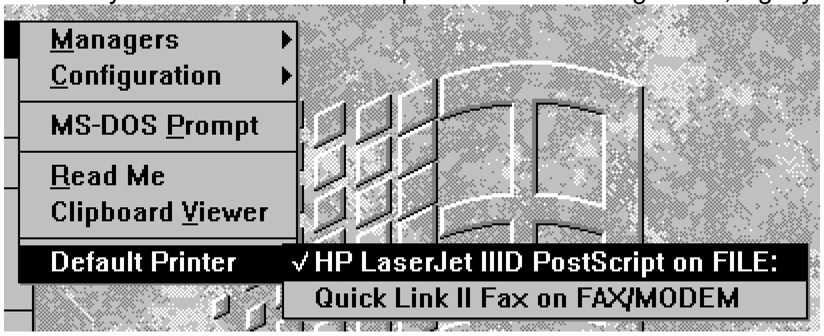
The name of the DLL within which the function resides.

The name of the function contained within <DLL Name> which is to be called.

The rest of the data after the function name is combined into a single string and passed to the function.

$DefaultPrinter in DEFPRN.DLL

This keyword presents you with a pull-right menu containing all of the available printers you have currently installed. One of these printers has a tick against it, signifying it is your current default printer.



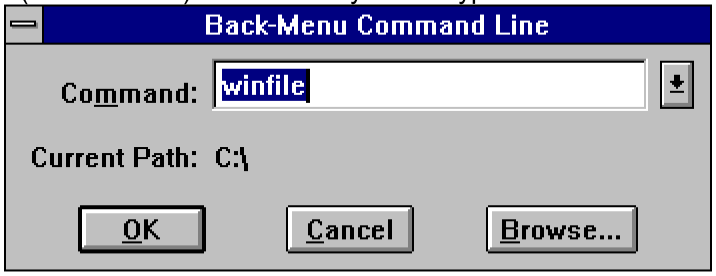
Choosing any printer from the list will make it your default printer.

$EditMenu

Use the editor defined using $SetOptions to edit the current menu.

$Execute [Browse]

Want to run an application but it doesn't appear within your menu? This keyword provides a dialog box (shown below) within which you can type the command line to execute.



Command: winfile

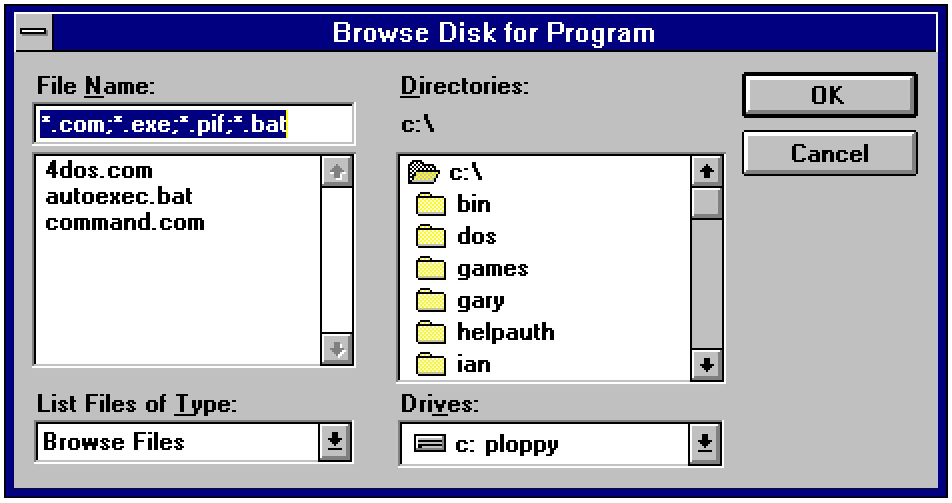
Current Path: C:\

Command

Type in the command line to be executed. A previous history of the last 10 commands is kept and can be accessed by clinking on the down arrow.

Browse...

Rather than typing in the location of a file, you can use the browse button to bring up a dialog box which can then be used to find the file.



File Name: *.com;*.exe;*.pif;*.bat

4dos.com
autoexec.bat
command.com

Directories: c:\

c:\
bin
dos
games
gary
helpauth
ian

List Files of Type: Browse Files

Drives: c: ploppy

Takes you straight to the dialog box for finding a file. It's just as though you choose **$Execute** and clicked on the **Browse...** button straight away.

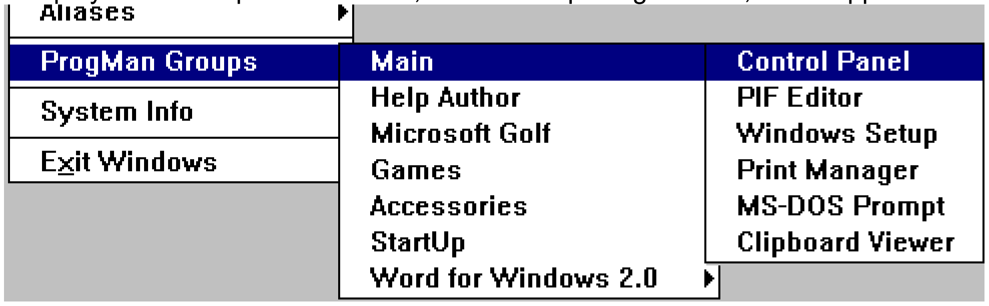
$ExitWindows [NoConfirm] in BMKEYW.DLL

Exit windows and return back to DOS (or whatever operating system you are using).

The **NoConfirm** option disables the warning message.

$Groups [Sorted]   in PMGROUPS.DLL

Provide a pull-right menu which contains all of the Program Manager groups. Each group will be displayed and will provide access, via another pull-right menu, to the applications contained within.



Note The menu and all sub-menus are built when BackMenu is loading the menu file. If you make any changes to the groups, these will not be reflected by BackMenu until the menu file is re-read. It is possible to force BackMenu to re-read the group files by executing the **$ReloadMenu** command, however this will only work if the **NoGroups** parameter is not supplied.

See Also

$ReloadMenu

The group descriptions, and all the entries within each group, will be sorted alphabetically.

$HideAlias [Selected]

Removes, but does not destroy, the aliases/tools from the screen. Useful for uncluttering your desktop.

See Also

$AliasToFront $AliasToBack $ShowAlias $ArrangeAlias $SnapAlias $NewAlias $SaveAlias $AliasOptions

$Info

Want to know lots of interesting facts about your Windows environment? Then look no further than this keyword. It will tell you more things than you ever wanted to know.

$LoadMenu [PopUp] <Menu File>

Replaces the current menu with one contained in the specified file.

E.g.

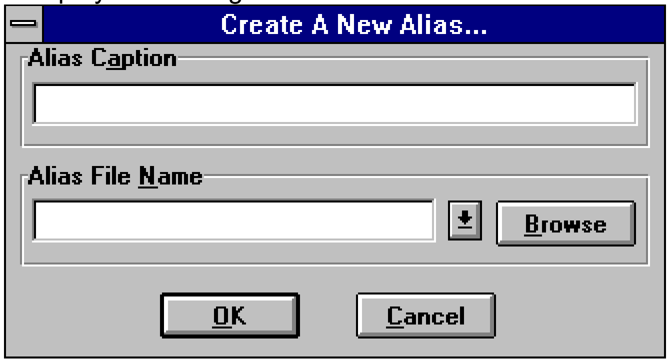
```
My Menu,$LoadMenu c:\users\ian\mymenu.ini
```

Causes the new menu to pop-up as soon as it has been loaded, rather than waiting for you to click on the background.

The DOS path and file name of the menu to be loaded.

$NewAlias

Displays the dialog box shown below which is used to create a new aliases.



See Also

$AliasToFront $AliasToBack $HideAlias $ShowAlias $ArrangeAlias $SnapAlias $SaveAlias $AliasOptions

General Information On Aliases

Alias Caption

The text that appears under the alias on the desktop. Be as descriptive as you like...

Alias File Name

The actual file name to be associated with the alias. This is then turned into a command to be executed when you double click on the alias. It can contain any or all of the attributes described in the BackMenu syntax summary. You can also associate any of the keywords as an alias command line. Click on the down arrow to get a list of them all.

An alias file name may contain a hash (#) character which denotes where in the command any files dropped on the alias are to appear. This is more fully described in the BackDrop alias/Tool summary.

$ReloadMenu [NoGroups] [FreeDLLs]

The menu defined using $SetOptions will be re-read and replace the current menu. All of the Program Manager group files will be re-read if the $Groups keyword is encountered within the menu.

See Also

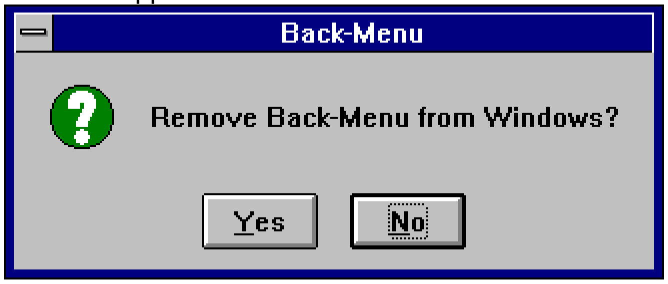
$LoadMenu

The Program Manager group files will be **not** be re-scanned, BackMenu will use the internal representation it creates when it is first run.

Any DLL references made using $CallDLL will be freed and the libraries are removed from memory.

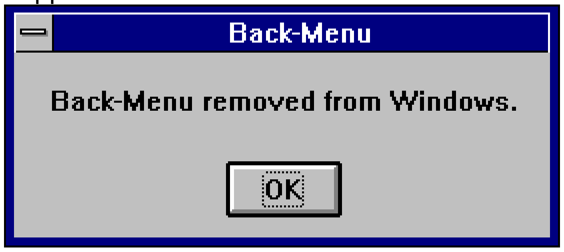
$RemoveMenu [NoConfirm]

Remove BackMenu from the Windows environment. All of the aliases and tools will be removed from the desktop and the menu destroyed.   You will be prompted to confirm that you really want to remove such a useful application.



Remove Back-Menu from Windows?

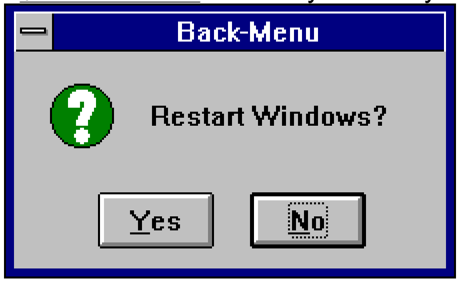
If you answer in the affirmative, BackMenu will no longer be available, and the following message will appear.



Back-Menu removed from Windows.

$RestartWindows [NoConfirm]

Want to shutdown Windows and then re-enter it so some changes you've made come into effect? Then **$ExitWindows** is the keyword for you. Answering yes to the question...


Back-Menu

Restart Windows?

will cause Windows to close down (just like **$ExitWindows)** and then restart.

$RunFile <file name>

Want to execute more than one command at once? $RunFile lets you do this. You create a text file which contains a list of commands, one per line, which are to be executed. The name of this file can then be given to $RunFile, and BackMenu will execute all of the commands listed.

E.g.

The file **`loadsfun.run`** contains:-

```
{XDESK=0 YDESK=2}c:\windows\sol.exe
{XDESK=1 YDESK=2}towers.exeTOWERS_SELL
{XDESK=2 YDESK=2}arachnid.exe
```

To run this little lot

```
Loads'o'Fun,$RunFile c:\backdesk\loadsfun.run
```

The full path and file name of the text file containing a list of commands to run.

$SaveAlias

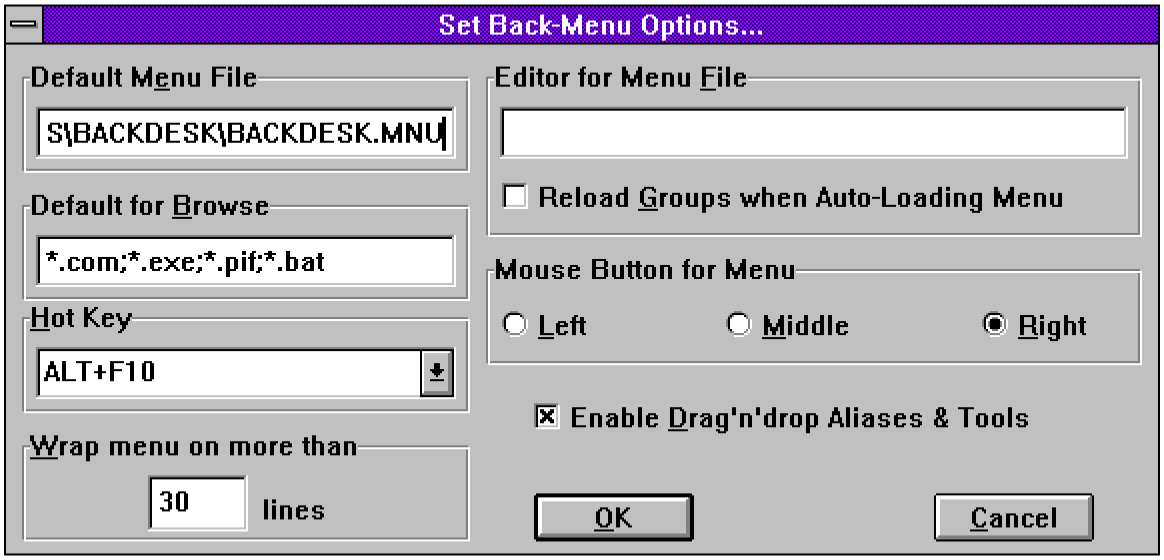
A small safety feature. This keyword stores the positions of all of the aliases and tools straight away. Useful if you've just spent a long time arranging things.

See Also

$AliasToFront $AliasToBack $HideAlias $ShowAlias $ArrangeAlias $SnapAlias $NewAlias $AliasOptions

$SetOptions

Lets you fiddle about with all sorts of settings to do with the secret innermost workings of BackMenu. Using this keyword will present you with a dialog box as shown below. Click on an item of interest for more information.



Set Back-Menu Options...

Default Menu File: S\BACKDESK\BACKDESK.MNU

Default for Browse: *.com;*.exe;*.pif;*.bat

Hot Key: ALT+F10

Wrap menu on more than 30 lines

Editor for Menu File:

☐ Reload Groups when Auto-Loading Menu

Mouse Button for Menu: Left Middle [✓ Right]

☒ Enable Drag'n'drop Aliases & Tools

Default Menu File

The name of the file containing the menu you want loaded when BackMenu starts up

Default for Browse

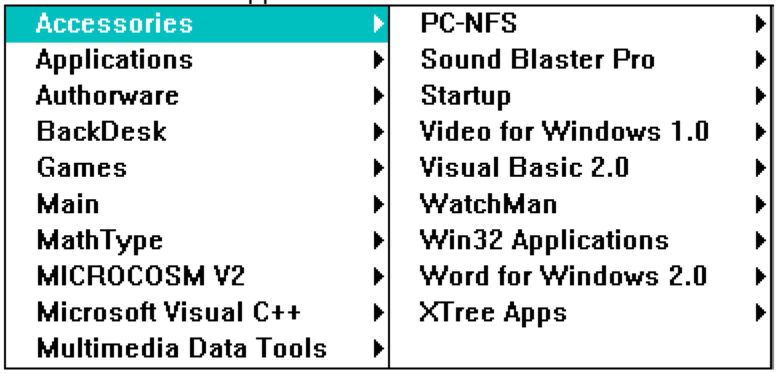
When using the **Browse...** button from any of the BackMenu dialogs, this string will be used to set the list of files initially displayed. You can edit this to include/exclude your favourite file types.

Hot Key

Choose which key pops-up the BackMenu menu. At the moment you're limited to ALT+F1 - ALT+F12.

Wrap menu on more than

Stop your long menu's from extending off the top and bottom of your display. This value defines how many lines a menu must contain before it is split vertically. For example, if the wrap value is set to 10, a 19 item menu will appear like this:-



Editor for Menu File

If BackMenu finds an error in your menu, or you activate the $EditMenu keyword, the application filename set here will be used to edit the menu. If its blank, then NOTEPAD.EXE is run.

Reload Groups when Auto-Loading Menu

BackMenu auto-detects if the menu has changed and will automatically load the new menu file. If this option is checked, it will also re-scan the program manager groups as well, otherwise it will use the previous values it found.

Mouse Button for Menu

Which button causes BackMenu to appear? You can select that here. If you're going to use the middle mouse button, make sure that the mouse driver you're running supports it.

Enable Dragndrop Aliases and Tools

BackMenu can provide aliases which act as shortcuts to your most frequently used applications. However, these aliases can interfere with other Windows Applications (such as Norton Desktop), so this check box allows you to enable (checked) or disable (unchecked) them. For more information on aliases and tools see the BackDrop Aliases and Tools section.

$ShowAlias

Re-displays any hidden aliases/tools. Useful for cluttering your uncluttered desktop.

See Also

$AliasToFront $AliasToBack $HideAlias $ArrangeAlias $SnapAlias $NewAlias $SaveAlias $AliasOptions

$SnapAlias [Selected]

Is the layout of your aliases and tools looking a little ragged? Want to line them up in neat rows and columns? **$SnapAlias** does all this, and no more. The grid used is defined using the settings contained within the BackDrop control panel

See Also

$AliasToFront $AliasToBack $HideAlias $ShowAlias $ArrangeAlias $NewAlias $SaveAlias $AliasOptions

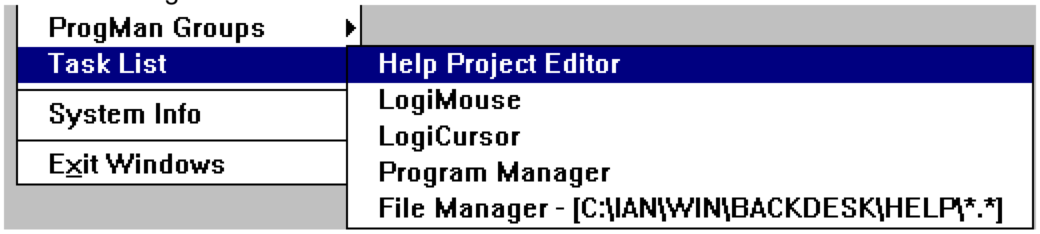
$StartSaver

If you've got a screen saver enabled, then this keyword will activate it. Useful if you want to lock your display in an unsecured environment.

Note	This keyword only works with screen savers accessed through the Control Panel. It will not work with other 3rd party savers such as After Dark.

$Tasks in BMKEYW.DLL

Provides a pull right menu which contains a list of all of the currently executing applications. If you select one of them, it will be brought to the front of all of the other windows and given the focus. An example of a task list is given below.

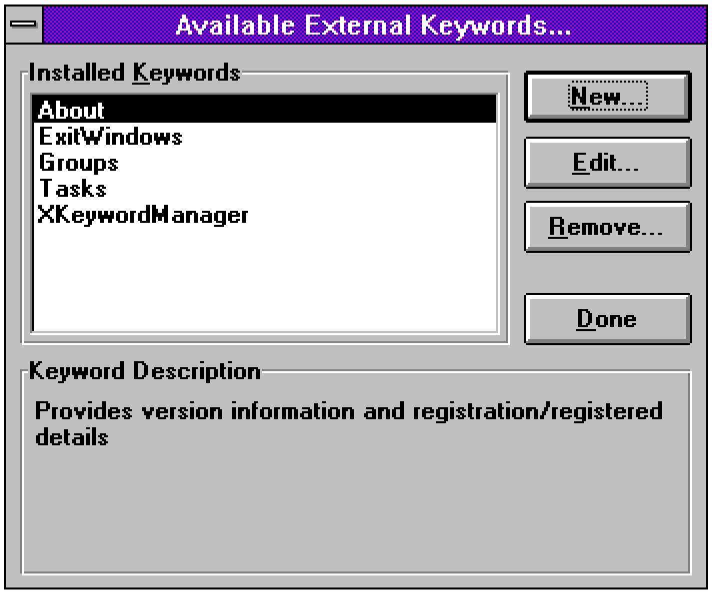


A Note about External Keywords

This keyword is an external keyword and, as such, may not be available if it has been uninstalled. For more information, see $XKeywordManager and the External Keywords section.

$XKeywordManager in XKEYMAN.DLL

This keyword is used to add, edit or remove any of the external keywords installed as a part of Backmenu. On executing it, you will be presented with a dialog box similar to the one shown below. Please click on an area of interest:-



Provides version information and registration/registered details

Installed Keywords

This list shows all of the installed external keywords available to use in BackMenu. You may select one or more of these to edit or remove. Selecting an individual keyword will cause its description to appear in the panel below.

Keyword Description

The description (supplied by the external keyword library) of the currently selected keyword.

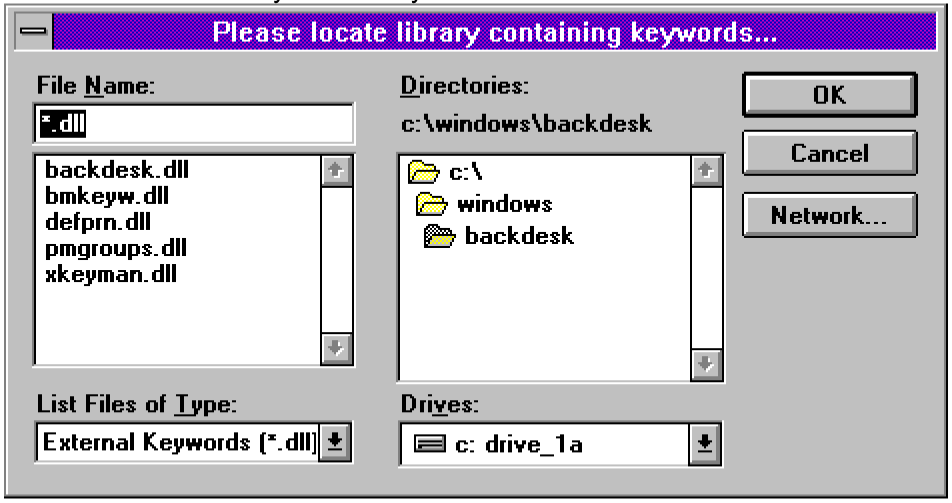
Available Keywords from Library

This list shows all of the external keywords it is possible to install from this library. You may select one or more of these to add. Selecting an individual keyword will cause its description to appear in the panel below.

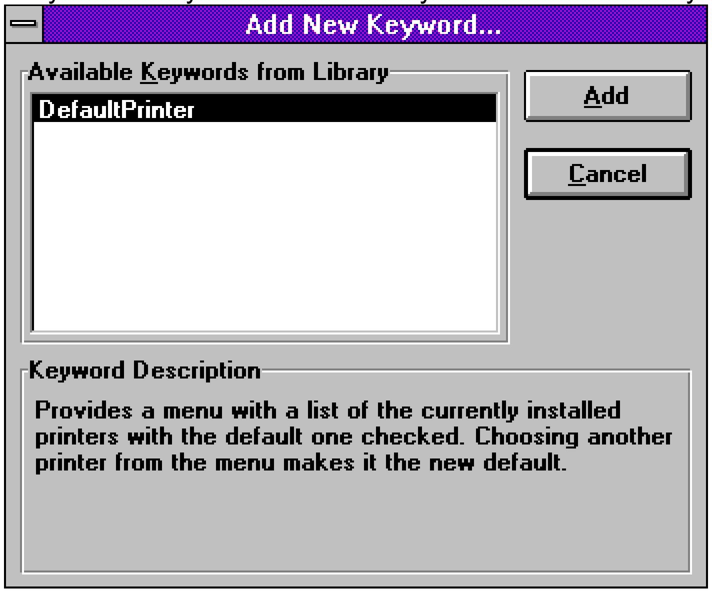
Add the selected keywords to the installed set.

$XKeywordManager - Installing a new External Keyword

Installing a new external keyword is simple. First of all, use the common dialog box shown below to locate the external keyword library.



Once youve located the library (and there are no problems), a dialog will appear listing the external keyword or keywords available to you. Choose the ones you wish to install and then click Add.

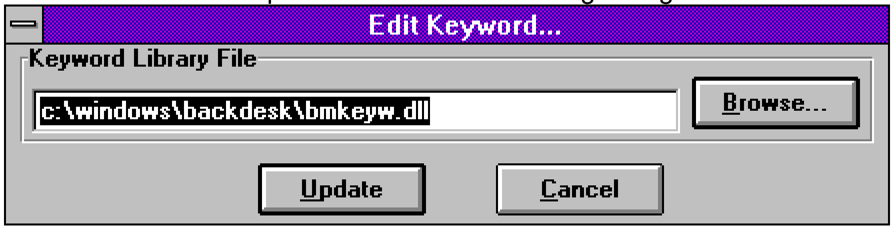


Provides a menu with a list of the currently installed printers with the default one checked. Choosing another printer from the menu makes it the new default.

You might get an message box containing an error message if something goes wrong during the installation process.

$XKeywordManager - Editing an existing External Keyword

Editing a keyword lets you change the name of the external keyword library which contains the keyword selected. You will be presented with the following dialog box.



c:\windows\backdesk\bmkeyw.dll

This button lets you search for the new location of the library containing the external keyword. It brings up a dialog box similar to the following:-

When youve edited the external keyword library, press this button to have BackMenu use your new settings.

$XKeywordManager - Removing an External Keyword

If you no longer want to use a particular external keyword, you can remove it from the set available and thus free up any resources it may have used. To do this, choose remove from the $XkeywordManager dialog box. You will be asked if this is something you really want to do.

Please note

1. Your menu will be reloaded and parsed as soon as you have finished with this keyword. If you have removed any external keywords and they still exist within your menu, BackMenu will report them as an error. We suggest you remove all occurances of any external keywords you wish to remove before you actually remove them through $XKeywordManager.
2. $XKeywordManager is itself an external keyword and therefore can be removed, thus preventing anyone from tampering with the set if installed external keywords. The only way to reinstall this keyword is to follow the instructions given in So how do I use my shiney new external keyword in BackMenu?.

BackDrop Aliases and Tools

This section describes how aliases and tools work within the BackDesk package. Although they are dealt with separately, you need to be running BackMenu in order to use them. BackMenu does all of the caring for the aliases, saving them when you quit BackMenu and restoring them again the next time you start it up.

BackDrop Aliases and Tools - What are they?

BackDrop Aliases and Tools - How do I use them?

BackDrop Aliases and Tools - Keyboard short cuts

Drag'n'Drop Tools supplied with BackDesk

BackDrop Aliases and Tools - What are they?

So, what is an Alias?

A BackDrop alias is an icon which appears on the desktop and represents a particular file on your machine. This file can be either an application program or a data file and you can use any icon you like to represent it. Aliases can be activated simply by double-clicking on the alias icon. If the alias is an application program, it will be executed. If the alias is a data file, BackDrop will use the file association (as set up using File Manager) defined for that file type to "execute" it. An error will occur if no association exists.

An alias can also be used as the source for a drag'n'drop operation. You can drag aliases and drop them on applications (or other aliases or tools, more later) in the same way you can drag files from File Manager. If an alias is defined correctly, you can also drag files from File Manager and drop them on the alias. In order for an alias to be a destination for a drag'n'drop, it must be an alias to an application program.

The alias definition can be thought of just like a command line. Normally the command line is simply the file name associated with the alias, in which case activating an alias is just like double clicking on the associated file in File Manager. Because it is a command-line, however, you can also use all of the BackMenu attributes available in your menu definitions (in fact, BackDrop just uses the BackMenu execution engine to actually do the work).

So, you can have an alias with a definition like:-

```
{XDESK=1 YDESK=1 XPOS=100 YPOS=100}c:\windows\system.ini
```

which, when double-clicked, causes Notepad (the default association with .INI files) to be run at (100,100) on desktop [1,1]. You can still also this alias and drop it on a running Notepad, in which case the Notepad will open the file c:\windows\system.ini.

Aliases can also have files dropped on them. To make an alias "drop aware", you use the hash (#) symbol in the alias command line. This hash is replaced with the file dropped on the alias and the resulting command is executed. So another alias might be defined as:-

```
{XDESK=1 YDESK=2}notepad.exe #
```

If this alias is double clicked, Notepad will be executed. If a file is dropped on the alias, the # is replaced with the file and resulting command executed. If the alias in our first example were to be dragged and dropped on this alias, Notepad would open on desktop [1,2] with the file c:\windows\system.ini. Note that the attributes for the original alias are lost, Windows drag'n'drop only allows file names to be passed to the application. It is even possible to drag this second example and drop it on another alias or application. If this were to happen, the application would try and open NOTEPAD.EXE (probably not what you'd want to happen, but its possible).

You can define what happens if more than one file is dropped on an alias at once (i.e. you select 4 files in File Manager and drop them on your notepad alias). Two things can happen; the alias command is executed once for each file dropped (so you'd get 4 Notepads running in our example). The alternative is to have *all* of the files replace the #, in which case one Notepad is run with all 4 files on its command line. Both of these modes of operation are available to each alias.

OK, but what is a Tool?

A tool is a simple application written to run in conjunction with the BackDrop alias system. It appears just like an alias, and operates in the same way. Any tools executed are saved along with the aliases by

BackDrop, and started up again the next time BackMenu is used.

The difference is that a tool is a running process and so can do more interesting things with the files dropped on it. For example, one of the tools supplied with BackDrop is a wallpaper changer. You can drop bitmap files on it and the background wallpaper will change. This tool does much more than this, it allows you to keep and manager separate wallpaper lists, and have your wallpaper change while you are using Windows. This behaviour would be impossible to do with a simple alias.

BackDrop Aliases and Tools - How do I use them?

The way in which aliases and tools can be used is summarised in the introduction. However, there are various operations you can perform on them which are summarised in the list below.

Creating a new alias or tool

Selecting aliases and tools

Moving aliases and tools

Configuring aliases and tools

Saving and loading aliases and tools

Closing aliases and tools

Disabling aliases and tools

BackDrop Aliases and Tools - Creating

There are three ways to create an alias:-

- By selecting a menu item using the $NewAlias keyword. A dialog box will prompt you for details of the alias.
- By dragging a file from File Manager and dropping it on the desktop. A default Icon is chosen for the alias depending on the type of file dropped.
- By using the T2Alias tool.

BackDrop Aliases and Tools - Selecting

An alias or tool can be selected (activated) in a number of ways. To do this, use the mouse button not assigned to bring up the alias/tool menu (see the BackDrop control panel for details on how to set this). You can:-

- Click on the alias/tool to select it. Depending on the mouse button chosen for the alias menu, a menu might appear.
- Shift click on an alias to add or remove an alias/tool from a collection.
- Click and hold the mouse button on the desktop (not using the mouse button defined to activate the BackMenu menu) and drag a rectangle around any aliases/tools. The aliases/tools whose centre falls within the rectangle are selected. Holding the shift key down while performing this operation will add the aliases/tools selected to the current selected set.

Once the aliases have been selected, you can perform group operations on them.

See Also

Moving Aliases/Tools Closing Aliases/Tools $AliasToBack $ArrangeAlias $HideAlias $SnapAlias

BackDrop Aliases and Tools - Moving

Once a single alias or tool (or group of them) has been selected, you can move them about. This can be done with either the keyboard or the mouse :-

- Cursor keys can be used to move the currently selected aliases/tools.
- Click, hold and dragging on an alias/tool will move the currently selected set.

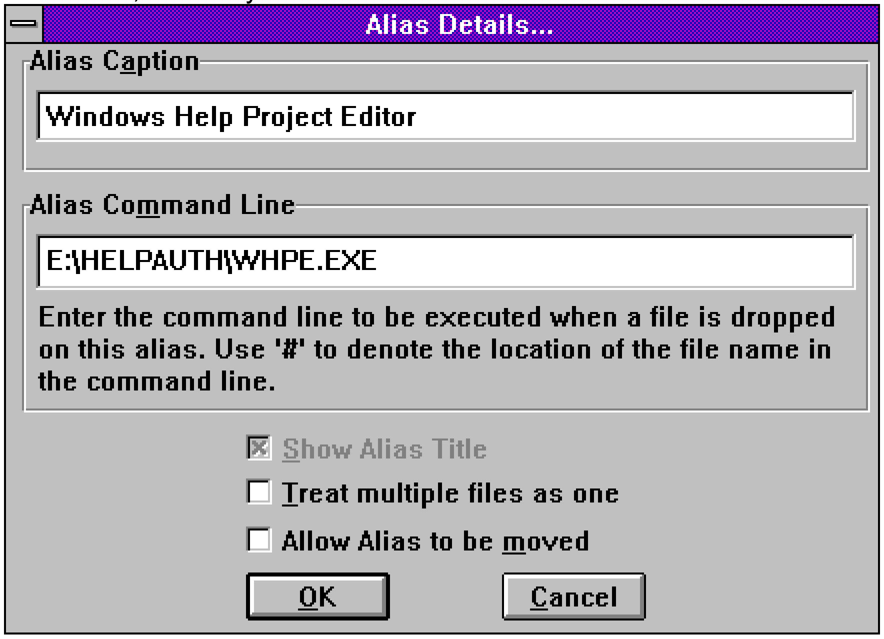
See Also

Selecting Aliases/Tools

BackDrop Aliases and Tools - Configuring

Aliases can be configured by using their menu. Click on the alias using the button defined to activate the menu (see the BackDrop control panel for details on how to set this). Each tool has its own menu, see the help for that tool for details on how to use it.

There are two dialog boxes available to configure aliases. The first, displayed by selecting "Alias Details...",   allows you to define how the alias works.


Alias Details...

Alias Caption: Windows Help Project Editor

Alias Command Line: E:\HELPAUTH\WHPE.EXE

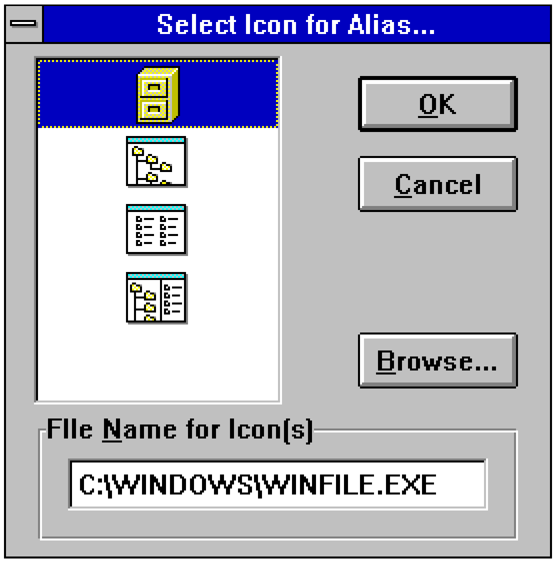
Enter the command line to be executed when a file is dropped on this alias. Use '#' to denote the location of the file name in the command line.

[✓ Show Alias Title]

☐ Treat multiple files as one

☐ Allow Alias to be moved

The second, displayed by selecting "Alias Icon...", allows you to choose the icon used to represent the alias.


Select Icon for Alias...

FIle Name for Icon(s): C:\WINDOWS\WINFILE.EXE

Alias Caption

The text displayed under the alias. You can remove the caption from an alias in one of three ways :-

- Leave this entry blank.
- Turn off the "Show Alias Title" setting in this dialog box.
- Disable the captions for all aliases via the $AliasOptions configuration dialog box

Alias Command Line

The command to be executed when the alias is double-clicked. This normally contains the name of the file used to create the alias, but you can use any part of the BackMenu command line syntax (as described in the introduction).

Show Alias Title

Check this option if you want the caption to appear below the alias icon. If you have disabled all alias captions via the $AliasOptions configuration dialog box, this option will be greyed out.

Treat Multiple Files As One

This option decides what happens when you drop more than one file onto an alias. Normally, the alias command line is executed with the hash character replaced for each file dropped. If this option is checked, however, the alias command line is executed once with the hash character replaced by *all* of the files.

Allow Alias to be moved

Setting Nail Alias/Tools to Desktop within the $AliasOptions dialog means aliases and tools stay put. Checking this option lets you override the global setting and fine tune the position of the alias. This option is always set for newly created aliases (i.e. from $NewAlias or from a drag/drop operation).

Shows you all of the icons available from the file displayed below. Select the icon you want to use and click the "OK" button.

Browse...

Displays a dialog box from which you can choose the executable or library file to look for icons in.

File Name for Icon(s)

Type in here the name of the file you want to extract the icons from. Click on "OK" and any icons found in the file will be displayed for you to choose from.

BackDrop Aliases and Tools - Saving and Loading

The position, icon, command-line etc. of an alias or tool is stored away in a file called SP-SERV.INI which is located in your Window's directory. Please feel free to have a look at this file, but don't touch anything! SP-Services applications use this file to store essential information, changing entries could cause unpredictable results.

The current set of aliases and tools are loaded whenever BackMenu is run, that's all there is to it!

Alias and tool information is *always* saved if you execute the $SaveAlias keyword. You can also tell BackMenu to save these settings whenever you quit the application (or exit Windows). BackMenu can be set up to prompt you (if you don't always want to save the aliases you've got), or even not to save anything at all (see the $AliasOptions keyword for details on how this can be configured).

This last option may not seem to be useful, but is if you've gone to a lot of trouble to create the perfect set of aliases and tools. If you turn off the "save when quitting" option, this set will never change and is safe from accidental moving/deleting/changing (unless you use the $SaveAlias keyword). The flip side is you have to remember to explicitly save your alias and tool set-up whenever you change it - it's your choice.

BackDrop Aliases and Tools - Closing.

You can close an alias or tool by using *CTRL+ALT+X*. This applies to *all* of the currently selected aliases and tools so use these keys with care!

Note ALT+F4 does *not* remove an alias/tool.

BackDrop Aliases and Tools - Keyboard short cuts

You can alter the general layout and state of aliases and tools by using the keyboard. Most of these keyboard short cuts have equivalent BackMenu keywords. These keyboard short cuts can be grouped into two distinct classes.

- Those that affect only the selected aliases and tools (CTRL + key).
- Those that affect *all* aliases and tools (CTRL + ALT + key).

The table below shows the various keyboard short cuts.

Keyboard short cut	Function	Keyword Equivalent
CTRL+A	Arrange selected Aliases/Tools	$ArrangeAlias selected
CTRL+ALT+A	Arrange all Aliases/Tools	$ArrangeAlias
CTRL+B	Selected Aliases/Tools to back	$AliasToBack selected
CTRL+ALT+B	All Aliases/Tools to back	$AliasToBack
CTRL+ALT+F	Bring all Aliases/Tool to front	$AliasToFront
CTRL+G	Snap selected Aliases/Tools to grid	$SnapAlias selected
CTRL+ALT+G	Snap all Aliases/Tools	$SnapAlias
CTRL+H	Hide Selected Aliases/Tools	$HideAlias selected
CTRL+ALT+H	Hide All Aliases/Tools	$HideAlias
CTRL+ALT+R	Restore all Alias/Tool positions	Note 1
CTRL+ALT+S	Show all Alias/Tools	$ShowAlias
CTRL+ALT+X	Close *selected* Aliases/Tools	Note 2

Closing Aliases and Tools

Like all good rules, this one has an exception and that is *CTRL+ALT+X*. This exception allows you to close one or a whole group of aliases/tools in one stroke. As this is such a powerful command it needs a powerful set of keys to activate it. You're far less likely to hit CTRL+ALT+X than CTRL+X by accident.

Restore all Alias/Tool positions

If you inadvertently muck up your wonderful alias/tool layout, it is possible to repent. See BackDrop Aliases and Tools - Restoring their positions for more details.

BackDrop Aliases and Tools - Restoring their positions

Moved all of your aliases and tools around and decided the layout is no good? Inadvertently dragged an alias instead of clicking on it? Chosen to arrange the aliases when you really meant to snap them to the grid? If any of these situations arise then the restore keyboard shortcut is designed for you. Simply press *CTRL+ALT+R* and the aliases and tools will be restored to their last remembered position.

How does BackDrop decide when to remember the positions of all of the aliases and tools? If you leave them alone (i.e. don't move them) for a period of about 30 seconds, BackDrop decides that you're happy with the layout and so remembers it.

Note Currently there is no keyword equivalent to this command, it only works from the keyboard.

BackDrop Aliases and Tools - Enabling and Disabling

We hope you will find the aliases and tools a useful addition to your desktop. However, you may wish to disable this feature for some reason (e.g. it conflicts with another Desktop application). To do this, un-check the Enable Dragndrop Aliases and Tools option from the $SetOptions dialog box. If, at some later stage you wish to use aliases and tools again, simply check this option.

Note BackMenu will have to be restarted before this change can take effect. If youve installed BackMenu as the shell, youll have to restart Windows. BackMenu will inform you of the correct course of action via a message box.

An Overview of BackMenu Command Line Syntax

This section describes just what you can place on a BackMenu command line. This command line can appear in many places within BackMenu.

1. As part of the definition of a menu.
2. As part of the definition of an alias.
3. In a text file grouped with other commands, to be executed together via **$RunFile**.
4. As an entry in the dialog box accessed via **$Execute**.

Most of the time, the command line used within BackMenu will look exactly the same as commands entered at the DOS prompt. However, because BackMenu is executing Windows programs, it can have a lot more control over exactly how an application starts up.

A BackMenu command line can consist of 3 parts:-

1. An optional set of attributes which are applied to an application and are contained within curly braces **{}**
2. The name of the application to be run
3. Any parameters that are to be passed to the application.

E.g.

```
{NOR XDESK=2 YDESK=2 XPOS=10 YPOS=10}NOTEPAD.EXE hello.txt
```

Click on the part of the command line you would like more help with.

You can use this command line syntax from other applications (e.g. Program Manager) by using BDX, supplied with this package.

The Command Line Attributes - Overview

The BackMenu command line attributes give you complete control over how the application starts up. You can:-

Specify start-up state, size and position

Control the virtual desktop

Specify the default directory

Run a single instance only

Delay before moving and sizing

Stop BackMenu interpreting the application parameters

These attributes are placed on the command line before the application program name and enclosed in curly brackets **{}**. The general format for an attribute is

```
NAME=numeric-value
```

or

```
NAME="string"
```

Only the first few characters of an attribute are required to differentiate between them. In the following examples, the minimum number of characters required are shown in capital letters.

Please click on the list above for more information about the various attributes.

The Command Line Attributes - Start-up State, Size and Position

When executing an application, BackMenu gives you complete control over how and where it will appear on the screen. You can specify an application's:-

• Position	{**XPOs** and **YPOs**}
• Size	{**WIDth** and **HEIght**}
• State	{**NORmal**, **MAXimized**, **MINimized**, **HIDden**}

These attributes can be mixed and matched. For example, to start up Notepad at location (100,100) on the screen, you might execute:-

```
{XPOS=100 YPOS=100}NOTEPAD.EXE
```

To also make this window 200x300 you would execute:-

```
{XPOS=100 YPOS=100 WIDTH=200 HEIGHT=300}NOTEPAD.EXE
```

The Command Line Attributes - Controlling the Virtual Desktop

If you're running BigDesk, then you can also use the following attributes for extra control over how an application starts up. The attributes **XDEsk** and **YDEsk** specify which desktop the application will use. Desktops are numbered from 0 from the top-left corner.

For example, to start up Seahaven Towers on the middle desktop of a (3x3) set-up you would use:-

```
{XDESK=1 YDESK=1}TOWERS.EXE
```

These can be used alongside the normal positioning controls, or instead of. For example, you can specify screen position as well as the desktop. In this case the window will be placed at the required position relative to that desktop. For example:-

```
{XDESK=2 YDESK=0 XPOS=0 YPOS=0}MINES.EXE
```

will be placed in the very top-left hand corner of the specified desktop. It is possible, by using large or negative screen co-ordinates, to position a window so it straddles several desktops. If you don't want this to happen, you can specify the **SNAp** attribute which ensures the whole window (or as much of it as possible) is on the specified desktop.

You can also make use of the other facilities available through BigDesk, such as the keep-to-front and keep-with-desktop options, which can be set for a particular application. BackMenu supports two attributes, **TOFront** and **TODesk**.

TOFront makes the application float above all other windows (except those which are themselves TOFront).

TODesk makes an application stay with you when you move from desktop to desktop, rather than remaining in its specified place.

For example:-

```
{XPOS=0 YPOS=0 WIDTH=100 HEIGHT=100 TOFRONT TODESK}CLOCK.EXE
```

will execute a clock in the top-left hand corner of the screen, that will float above all other windows and will remain with you whatever desktop you are on.

A very addictive single pack patience game also available from Clarity Systems :->

The Command Line Attributes - Single Instance Applications

There may be instances when you want to execute something once and once only. If the application is already running you'd rather switch to it than start up a new copy. This may be because the application can only have one copy running at a time (e.g. BigDesk). Or, it mat be simply that you don't want to clutter your desktop with multiple copies of the same thing.

A good example of this is using a particular help file. Each time you select the help file from BackMenu, a copy of WINHELP will be run with the help file displayed in it. It would be better to switch to the window containing the help if it exists, and only execute it if not.

BackMenu lets you perform the above feat by providing the **SINgle** attribute. This tells BackMenu to look and see if there is another copy of the application running before it tries to execute it. However, for BackMenu to perform this feat, you must tell it how to identify another copy of the application. This is done by using the **TITle** and/or **CLAss** attribute along with SINgle.

With the TITle attribute you can specify all (or part) of the title of a window of the application to look for. BackMenu will examine all of the currently available windows and see if any start with the title given. If it finds one, it will make that window active then give up. Otherwise, will execute the command. The CLAss attribute works in a similar manner except the entire class name must be specified. CLAss is only provided for power users who like nothing better than rolling their sleeves up and rummaging around below the surface. Most of the time the TITle attribute will suffice.

For example, if you wish to execute only one copy of the Windows 3.1 SDK help file:-

```
{SINGLE TITLE="Windows 3.1 SDK"}WINHELP.EXE c:\windev\help\win31wh.hlp
```

The Command Line Syntax - The Default Directory

Occasionally, you may want an application to start up using a particular directory as default. This directory would be presented to you the first time you opened a document. By default, BackMenu uses the directory of the application, however, it is possible to override this using the **DIRectory** attribute.

For example, you have all of your spreadsheet documents in your own personal directory. To execute Excel and have it use that directory you might use:-

```
{MAX DIR=c:\ian\excel\finance}EXCEL.EXE
```

The Command Line Attributes - Delaying the Use of Attributes

BackMenu provides a powerful attribute set with which you can specify position/size/state of an application. However, there are applications which remember their own size/position information which can lead to problems,

A prime example of this is the support for running DOS based applications in a window.   There is an option in the Fonts... dialog box that makes Windows save the size/position of that DOS box on exit. If we now try and position this DOS box using   BackMenu attributes, things start to go wrong.

BackMenu executes the DOS program and re-locates the relevant window. Windows then comes along. notices the settings have been saved from before and so uses those to re-position the DOS box. A tug-of-war develops which BackMenu quickly loses. The DOS box appears where you last moved it and *not* where you wanted it to. This problem is compounded if you run several copies of a particular program; they all appear on top of one another!

To try and solve this problem, the **DELay** attribute was introduced. This attribute, surprisingly enough, delays the application of all other attributes by a certain amount of time, specified in milliseconds. This gives Windows (or the application) time to think it has positioned the window before BackMenu sneaks in and re-positions it. You may think that this is anti-social behaviour, but there are only certain applications where this attribute is required and you don't have to use it if you don't want to :->

To execute two DOS prompts at different locations, the following commands might be executed:-

```
{DELAY=500 XPOS=0 YPOS=0}COMMAND.COM
{DELAY=500 XPOS=0 YPOS=300}COMMAND.COM
```

The Command Line Attributes - Literal application parameters

BackMenu is able to process the parameters of an application and prompt for files if you specify wild-cards (for more information, see The Application Parameters - Prompting for files). However, there may be times when you wish to parse these parameters untouched into the application itself. To do this, specify the **LITeral** attribute, which tells BackMenu to leave the application parameters alone.

The Application Program.

This part of the BackMenu command line normally contains the file name (possibly with a full path) of the application which is to be executed. It is, however, possible to place the name of a document file here instead. Windows provides the ability to make associations between applications and particular file name extensions. For example, Notepad is associated with any file ending in .txt. This information is used in applications such as File Mangler, which lets you double-click on a file and execute the application associated with it. BackMenu provides the same service, For example:-

```
    {XPOS=100 YPOS=100}c:\winapps\backdesk\readme.txt
```

will cause Notepad (or whatever else has been associated with .txt files) to run and display the file name given. If no association exists, then the command line will fail to execute.

The Application Parameters

Application parameters are passed directly to the executing application, which then process them in some manner. For example :-

```
    NOTEPAD.EXE c:\windows\win.ini
```

causes Notepad to be run and load in win.ini from the windows directory. BackMenu provides some extra features that allow you to modify the application parameters before passing them to the application. These features are:-

Prompting for the application parameters

Prompting for files

The Application Parameters - Prompting

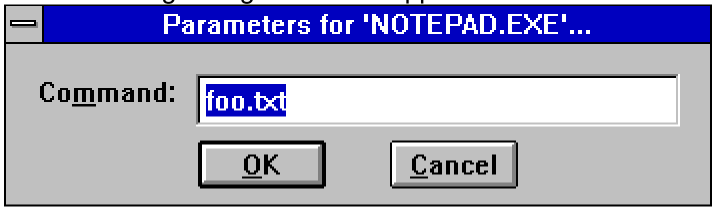
BackMenu lets you prompt for the entire application parameters. To do this, you place a percent (%) or ampersand (&) character at the start of the application parameters. BackMenu will then prompt with a dialog box containing any text which follows the % or &.

%	The text is readable.
&	The text is protected (comes out as ***). This is for security (e.g. prompting for a password).

For example, if you were to execute the following:-

```
{XPOS=40 YPOS=50}NOTEPAD.EXE %foo.txt
```

The following dialog box would appear:-


Parameters for 'NOTEPAD.EXE'...

Command: foo.txt

You can then edit the application parameters to your heart's content before clicking on OK to have those parameters used.

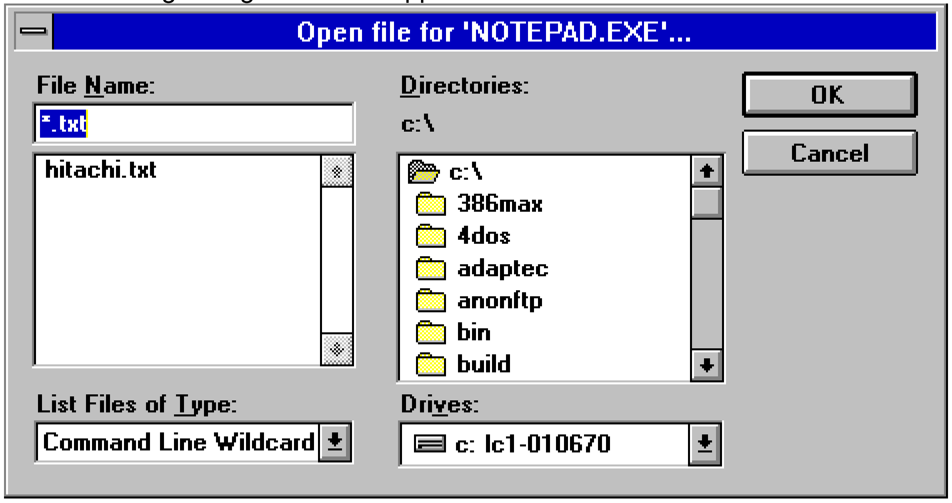
The Application Parameters - Prompting for files

BackMenu lets you prompt for files in the application parameters. To do this, use the standard DOS wild cards (* and ?). BackMenu will prompt with a dialog box which will allow you to wander the disk looking for the file of your choice. Note that this only works if the LITeral attribute has not been specified. If it has, the application parameters, along with any wild-cards, will be passed to the application.

For example, if you were to execute the following:-

```
    {XPOS=40 YPOS=50}NOTEPAD.EXE *.txt
```

The following dialog box would appear:-


Open file for 'NOTEPAD.EXE'...

File Name: *.txt

hitachi.txt

Directories: c:\

c:\
386max
4dos
adaptec
anonftp
bin
build

List Files of Type: Command Line Wildcard

Drives: c: lc1-010670

BackDesk V3.10

Welcome to the wonderful world of BackDesk!

BackDesk is a suite of utilities to enhance your Windows environment and make your work easier. BackDesk is not free; it is distributed as Shareware, allowing you to try before you buy.

Help is available on:-

- What's New in Version 3.10
- BackMenu - the pop-up menu
- BigDesk - The virtual desktop
- BackDrop - aliases and tools on the desktop
- BME - A new Windows editor for your BackMenu menu.
- WRunServ/WRun - Run Windows programs from a DOS box.
- WCopy/WPaste - Access the Windows clipboard in a DOS box.
- Shareware - How to do the decent thing and register your copy
- Known Problems and how to get support

Hey, I just got an error dialog!

Some applications running as icons, e.g. Program Mangler

Another Window's program, e.g. Notepad

This is BigDesk, the virtual desktop utility

This is BackMenu, the pop-up root menu utility

These are Aliases using the BackDrop facilities

This is the Desktop or Background on which the wallpaper is displayed

BackMenu does not appear to be running.   Click below to start it, then use the BackMenu button on the toolbar again to pop-up the BackMenu.

LAUNCH BACKMENU!

BigDesk does not appear to be running. Click below to start it.

LAUNCH BIGDESK!

Understanding DOS is left as an exercise for the reader.

BackDesk - What's New in Version 3.10

V3.10 of BackDesk has several new features/enhancements as well as many bug fixes. The major new enhancements are:-

- Installable external keywords
- Disabling DragnDrop aliases/tools (to work with other desktop applications)
- An Uninstall option
- You can Nail aliases and tools to the desktop (see $AliasOptions for details)
- You can use the BackDesk command line syntax from other applications using BDX.
- You can edit your menu using BME - a WYSIWIG menu editor
- BackMenu is NT Friendly. See the section Running BackDesk under Windows NT

New Keywords in 3.10

A few new keywords have been implemented.

$DefaultPrinter	Displays the printers available and allows you to select the default.
$Net...	A set of network keywords that allow you to add/browse/delete network resources.
$XKeywordManager	Allows you to add/edit/delete the external keywords available within BackMenu

New Attributers in 3.10

- LITeral stops BackMenu from interpreting any wild-cards (* and ?) it finds in the application parameters

Bug Fixes in 3.10

- BigDesk will now cope if you change video resolution while it is running
- If you use wild-card parameters on a BackMenu command line and press cancel, a spurious error is no longer given.
- Numerous other bug-fixes which Ive forgotten

As well as these additions, all of the original features which make BackDesk V3.xx great are still present with one exception. The system menu entries, which allowed you to activate BackMenu and/or BigDesk from the system menu in any other application has been removed - it caused too many problems with other applications. Sorry!

- List of supplied files
- Drag'n'Drop Tools

BackMenu

The syntax used within BackMenu has been revised as we were running out of strange characters to use as tokens. Instead the syntax uses an attribute system, where the attributes and any associated values are placed at the start of the command. The new syntax is:-

@{attribute List}Command parameters

The @ is used, as before, for auto-start and is optional.

The command and parameters are self explanatory.

The attribute list allows you to modify the manner in which the application is run. With them you can specify the size and position of the window, the start-up directory or any number of other things. The current set of attributes is described in BackMenu - Command Line Syntax

New Keywords in 3.00

A few new keywords have been implemented.

$AliasOptions	Change BackMenu alias settings
$AliasToBack	Send the aliases/tools behind all other windows
$AliasToFront	Bring the aliases/tools above all other windows
$ArrangeAlias	Arrange the aliases along one side of the screen
$EditMenu	Edit the current menu.
$HideAlias	Remove all aliases/tools from the screen.
$LoadMenu	Load in another menu-file to replace the current.
$NewAlias	Create a new alias
$RestartWindows	Restart Windows
$RunFile	Execute a set of commands contained within a file
$SaveAlias	Save all the alias/tool positions.
$ShowAlias	Bring all aliases/tools back onto the screen
$SnapAlias	Position aliases using a grid.
$StartSaver	Start the Windows screen saver

And a few keywords have been enhanced

$Execute	Choose an application to execute
$Groups	Display a menu containing the Program Manager Groups
$SetOptions	Set various BackMenu options

BackDrop - Drag and drop aliases

Aliases may be made between files in the system and icons displayed on the desktop. These icons have a raised look and italicised caption. These aliases are created by dragging files from an application that is a Windows 3.1 drag'n'drop source (e.g. File Manager) and dropping them onto the desktop. An alias icon is created using any association information or an icon from an executable. These aliases are stored as part of the BackMenu settings and so persist from session to session.

Once created, an alias may:-

- Be moved around the screen. Simply click-and-drag the mouse over the alias to move it. Multiple aliases can be dragged as one, by using SHIFT+Click to select more than one.
- Be double clicked. This executes the file associated with the alias. For example, an alias for NOTEPAD.EXE will execute Notepad when double clicked. Aliases for data files rather than executables will use the associations defined using File Manager. For example, double clicking on an alias for WIN.INI will start notepad with WIN.INI loaded.
- Be dropped on from File Manager. This executes the file associated with the alias, as with double clicking, but also passes the name of the file dropped as a command line. If the file is an executable, the dropped file will be passed as a command line parameter and the program should start up with that file loaded. For example, dragging SP-SERV.INI from File Manager and dropping it on an alias for NOTEPAD.EXE will start up Notepad with the SP-SERV.INI file loaded.

- Be dropped on from other aliases. Exactly the same rules apply as for files from File Manager, as each alias is an indirection to a file.
- Be dragged and dropped onto other executing applications. Any application that already understands the Windows 3.1 drag'n'drop protocol may have aliases dropped onto it in exactly the same way that files dragged from File Manager may be dropped. For example, dragging an alias for WIN.INI and dropping it onto a Notepad application will cause Notepad to load in WIN.INI.

Each alias may also be configured. A menu allows you to change the alias details or icon. The alias can also be closed and thus destroyed. Note that closing an alias does **not** delete the associated file, but only gives up some screen real estate.

The icon for an alias may be changed by selecting a file containing one or several icons and then choosing one. The icon may be in an executable, dynamic link library or icon file. If no icon can be found, a default is used. The name of the file and icon number are stored as part of the alias. Move the file and BackMenu will no longer be able to locate the icon and so it will revert to the BackDrop one.

The alias details dialog box allows you to change the caption for the alias and the associated file. This is shown as the alias command line with the file as the first item. By changing this item, the file associated with the alias is changed. You can also modify the command line, by adding in parameters etc. There is a special symbol (#) which denotes the location in the command line of a file if one is dropped onto this alias. E.g. the alias command

```
C:\COMMAND.COM /C COPY # a:
```

will expand into

```
C:\COMMAND.COM /C COPY FOO.TXT a:
```

if the file FOO.TXT is dropped onto this alias. Note that the command line may contain any of the attributes described above.

BigDesk

You can drag a file from any Windows 3.1 drag'n'drop source (e.g. File Manager or a BackMenu alias) and drop it onto the virtual desktop. The program will be started up at the location dropped. Simple as that really.

The Save/Load Window Positions item from the Desktop menu allows you to save the positions of all executing applications and restore them later. This option generates a file called BIGDESK.DTF (for **D**esk **T**op **F**ile) in the same directory as BIGDESK.EXE. There's also an 'expert system' which uses a file called BIGDESK.RUL (for RULebase) to decide what items to store in .DTF files and how they should be stored.

CTRL+Left click on the background of the window when the title bar is removed and the system menu will appear.

BackDesk Package Contents

The files included in this package are:

BACKMENU.EXE	Pop-up menu
BIGDESK.EXE	Virtual desktop
BACKDESK.HLP	This help file
BDUNINST.EXE	A simple "uninstall" program
Required file for BigDesk:	
BIGDESK.RUL	The default rule-base for BigDesk
Required files for BackMenu:	
PMGROUPS.DLL	Program Manager group file reader
SAMPLE.MNU	Sample pop-up menu file
AN.DLL	External keyword manager
BMKEYW.DLL	External Keyword Library containing system keywords
Required files for any part of BackDesk:	
BACKDESK.DLL	Back-end engine for BigDesk and BackMenu
BACKDROP.DLL	Drag'n'drop library
BDCONFIG.CPL	BackDesk control panel applet
BDCONFIG.HLP	Help for BDCONFIG.CPL
CTL3D.DLL	Flashy 3D dialog stuff from MS
Optional files - BackDrop tools:	
WALLPAPR.EXE	Change your wallpaper every few minutes
DROPCLIP.EXE	Stack-up multiple files for a single drop
DUSTBIN.EXE	Drag'n'drop dustbin
T2ALIAS.EXE	Create an alias from a running program
COPYTOOL.EXE	Gather files into a single destination
Optional files - external keywords:	
DEFPRN.DLL	Choose the default printer from a menu
XK_NET.DLL	Add/Delete/Browse the network from a menu
Optional files - DOS tools:	
WCOPY.EXE	Copy a file/stdin to the Windows clipboard
WPASTE.EXE	Paste the Windows clipboard to stdout
WRUN.EXE	Run a Windows program from a DOS box.
WRUNSERV.EXE	Windows server required for WRUN.EXE
WINSTART.EXE	Stub for programmers incorporating WRUN
Optional files - other:	
POPGROUP.EXE	Display a .GRP file as a pop-up menu
BDSHELL.EXE	BackDesk shell stub to avoid Windows "feature"
README.TXT	Introductory stuff for the "GUI-challenged"
BDX.EXE	Execute a BackDesk command line from another application
BME.EXE	A WYSIWYG menu editor

Drag'n'Drop tools

The drag'n'drop system also allows tools to be written, which may then interact with the rest of the drag'n'drop environment. This appear as icons on the desktop in the same manner as aliases, except the caption is not italicised. The tools act in the same manner as aliases. Each can have files or other aliases dropped onto them. The tools, however, are windows programs and perform some set function with these files. Tools may also be saved along with aliases and so can be automatically reloaded when the system is started once more. BackDesk comes with five of these tools:-

- **WallPaper** is a tool that accepts bitmap files (by dropping) and then cycles through each as the background wallpaper. The delay between changes can be set in minutes. You can fiddle with settings for each wallpaper (i.e. whether it is tiled or centred).
- **DropClip** allows a batch of files to be "saved up" over a period of time, and then dragged onto another application. This tool is useful if you want to drag files from File Manager in one desktop to another program in another desktop. As the virtual desktop does not dynamically scroll, the files can be dragged and dropped into DropClip, the desktop focus moved, and dragged out of DropClip and into the other program.
- **Dustbin** is a drag'n'drop file and alias remover. Simply drag the aliases or files that are to be removed and drop them onto the dustbin icon. Note that files dropped into the dustbin are irretrievably deleted.
- **TaskToAlias** is a tool to create an alias from a program that's already running. Double click on it and you'll get a spyglass cursor. Position this over any other window and click and an alias will be created for you from the details of the program over which you clicked.
- **CopyTool** is a tool to help you gather together a set of files for copying to somewhere else. Drag any file from a program like File Manager, drop onto CopyTool. Set the options in CopyTool to tell it where and when to copy the files.
- **PopGroup** is not really a tool but there's nowhere else to tell you so... This program pops up a menu listing the contents of the Program Manager group file (.GRP) supplied to it as a command line parameter.   Associate it with the .GRP extension in File Manager.

Help Error Dialogs

If you just got an error dialog saying "Routine not found", it means that WinHelp can't see BACKDESK.DLL - you probably moved something! Don't worry, it just means that the BigDesk and BackMenu buttons above won't work.

Uninstalling BackDesk

We hope you will find BackDesk a useful and powerful addition to your Windows environment. However, we realise that there may come a time when you no longer wish to run our package so weve included a handy uninstall applet. Simply double-click on the Uninstall BackDesk icon within the BackDesk Program Manager group and BackDesk will be removed.

Running BackDesk under Windows NT

It is possible to use the 16 bit version of BackDesk on Windows NT as well as Windows 3.10 and Windows for Workgroups 3.11. At present the only limitations are :-

- BackMenu will not be able to show your Windows NT Program Manager groups. If you have installed Windows NT over Windows 3.10/3.11, BackMenu will display those groups, otherwise your $Groups menu entry is removed from the menu. We're working on this one
- Setting BackMenu to be your default shell will have no effect, you will always have Program Manager loaded.
- $ExitWindows and $RestartWindows will log you out, not exit or restart Windows NT.
- If you have full window dragging enabled, you may notice it takes time for the background to be repainted as you drag a window around. Don't worry, this is purely cosmetic and we're working on a solution.

All you have to do to make BackDesk work under NT is tell BackMenu. To do that, you simply place the -NT switch on the command line for BackMenu. You can do this by editing the BackMenu entry in the BackDesk 3.10 group. Select the icon in the group and choose File/Properties. In the edit box entitled 'Command Line' move the cursor to the end of the line and add -NT. Be sure there is a space between the end of the BackMenu application name and -NT. Choose OK, and that should be it.

If you run BackMenu without this switch, everything will still start up but you will not be able to display the BackMenu menu. The simplest way to remedy this problem is to log out and log back in again (this will close down the errant BackMenu, its the easiest way to be sure). It is also a good idea to do this if you get the error '*Unable to create background window*'.

Please note this is late-breaking code added to BackDesk 3.10, so there may be problems we have not foreseen. Please drop us a line if you encounter any problems running BackDesk under Windows NT (see the Shareware section for details)

BigDesk

One of the best things about Windows is that you can run several programs at once. The only real problem with this (apart, perhaps, from excruciating slowness when there are 30 or so copies of some game running at the same time) is that the screen gets awfully cluttered. BigDesk is the answer to this problem, and we think you're going to like it once you get the idea.

Imagine that you had nine monitors attached to your computer, all hidden behind a wall. Imagine a one-screen-sized hole in the wall. Now imagine that you can move the hole to look at any of the screens or even at parts of several of them. That's the thing BigDesk is going to give you. In order to use BigDesk to the full you will probably also need a copy of BackMenu, to provide you with a pop-up program menu on each of these 'virtual' screens.

For more information on using BigDesk, choose one of the sections below:-

- Mouse Operation
- Keyboard Operation
- Menu Operation
- The 'Set Options...' Dialog Box
- Drag'n'Drop
- The Application Menu
- Desktop Saving and Loading
- The BigDesk Rule-Base
- Using BigDesk from External Applications

BigDesk - Mouse Operation

There are several different operations which can be performed with the mouse. These are summarised in the table below

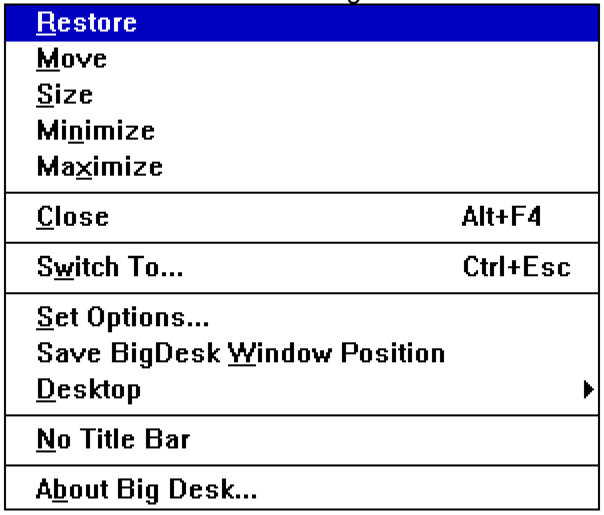
Mouse & Keyboard Operation	Where?	Effect
Right Click	anywhere	Move the virtual desktop to that location. If Snap Desktop to Grid is enabled, the desktop will be located to the nearest grid.
Right Click and Drag	anywhere	Drag the virtual desktop around. If Snap Desktop Grid is enabled, the desktop can only be moved in units of the grid size.
SHIFT+Right Click and Drag	anywhere	Drag the virtual desktop around. Overrides the Snap Desktop to Grid setting so the desktop can be placed *anywhere*.
Left Click	window	Brings that window to the front of all other windows
Left Click and Drag	background	Move the map window around on the real desktop (only if the title bar has been turned off).
Left Click and Drag	window	Move that window anywhere on the virtual desktop. The Window will be snapped to the upper left corner of a grid cell if the Snap To Grid option is set and the window is within the snap distance.
Shift Left Click and Drag	window	Move that window anywhere on the virtual desktop. The Window will be placed anywhere regardless of settings.
Left Click and Drag	background	Move the BigDesk window around on the Windows desktop, but only if the No Title Bar option is enabled.
Left Double Click	window	Activates that window and changes the virtual desktop so that window can be seen.
Left Double Click	icon	Restores that window to the current virtual desktop.
CTRL+Left Click	background	Brings up the BigDesk system menu if the No Title Bar option is set.
ALT+Left Click	window	Display a menu for that window from which you may perform both Windows and BigDesk operations.

BigDesk - Keyboard Operation

When BigDesk has the focus (i.e. is the window which is receiving all your attention at the moment & has the 'active' colour to its title bar), you can use the cursor keys (arrows, home, end, page up, page down) to navigate around the desktop.   The grid cell selector will move in the direction of the arrow pressed, with the other keys allowing diagonal movement.

BigDesk - Menu Operation

The menu operations are all to be found on the System menu.   To access this, click once on the icon or, with the map window open, click on the bar in the top left corner or press Alt+Spacebar. You can also get at the System menu by CTRL+Left click on the BigDesk window. For this to work, you need to click over the background of the BigDesk window **not** over any of the applications it is displaying. You'll see a menu which looks something like this :-



Standard System Menu Entries

We'll assume you can handle these. If not, please consult your Windows User Guide.

Set Options...

Allows you to tweak the various options in BigDesk. For more details, see BigDesk - Options.

Save BigDesk Window Position

Save the current size, position and state of the map.   Note that the identity, state and position of the applications which are executing *are not* saved, to do this see BigDesk - Desktop Saving and Loading.

About...

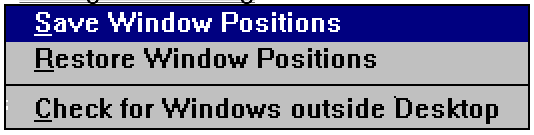
We'll leave you to explore this on your own.

No Title Bar

You can toggle BigDesk's title bar on or off by selecting this option. The system menu is still accessible (CTRL+Left click on the map background) and you can move the map window around (Left click and drag on the map background) with the title bar turned off.

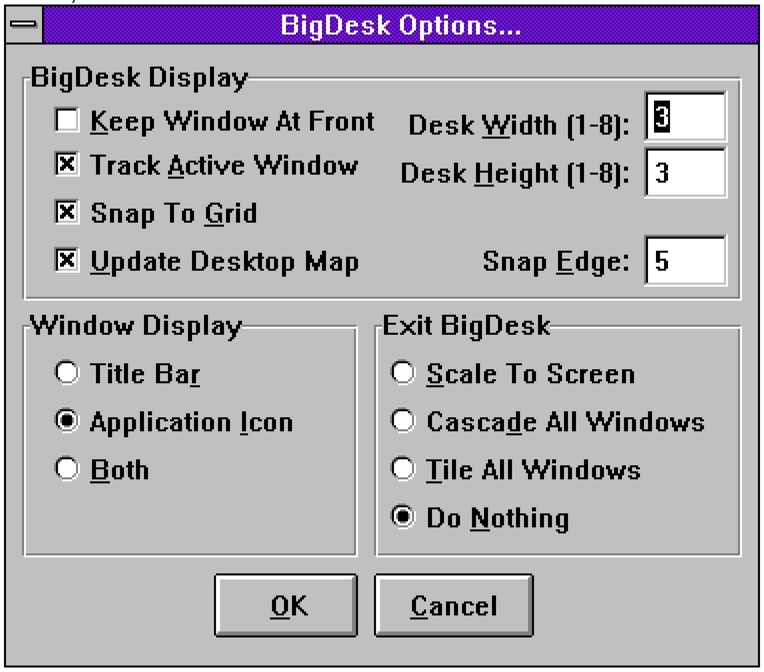
BigDesk - The Desktop System Menu

This menu lets you save and load the identities, sizes and positions of all of the currently executing Windows applications. For more information, click on a menu entry below or see BigDesk - Desktop Saving and Loading.



BigDesk - Options

Selecting the 'Set Options...' menu item from the system menu will produce the following dialog box. As ever, click for more information:-



BigDesk Display

- [ ] Keep Window At Front
- [x] Track Active Window
- [x] Snap To Grid
- [x] Update Desktop Map

Desk Width (1-8): 3
Desk Height (1-8): 3
Snap Edge: 5

Window Display

- ○ Title Bar
- ● Application Icon
- ○ Both

Exit BigDesk

- ○ Scale To Screen
- ○ Cascade All Windows
- ○ Tile All Windows
- ● Do Nothing

BigDesk - Keep Window To Front

Want to keep the BigDesk map visible all of the time? Then check this option in the 'Set Options...' dialog box. When it is enabled, BigDesk will keep its icon or map display on top of any window you may open. It may get in the way of something like word-processing, but is a very useful option if you are switching between virtual screens often.

BigDesk - Track Active Window

When this option is enabled in the 'Set Options...' dialog box, changing the window focus with Alt+Esc and Alt+Tab results in the newly selected window being moved to give the best view (all on screen if grid snap is off, in the cell with the largest proportion of the window if grid snap is on).

BigDesk - Snap To Grid

This option, set in the 'Set Options...' dialog box, places a grid over the entire virtual desktop, the size of each cell in the grid is that of the screen. You can think of this grid as a group of separate, easily accessible desktops - each one a simple right mouse click away.

The grid allows you to partition your windows up into groups (e.g. Word processors in the bottom left of the map, File Manager in the centre and Program Manager in the top right). By using the BackMenu attributes with the applications you execute, you can control how and where on this grid they are started.

If you turn off this option, the virtual desktop becomes one large area of which you can view any part by using the right mouse button.

BigDesk - Update Desktop Map

BigDesk will update its map of the desktop every second. Normally this is fine as you can see windows appearing as they are created. On some slower systems this may cause delays or slow response, so you can turn this feature off. If you do this, however, the map display will only update when you click on BigDesk.

To set this option, use the 'Set Options...' dialog box.

BigDesk - Virtual Desktop Width/Height

The 'Set Options...' dialog box allows you to decide how large the virtual desktop is to be. The units are in 'real desktops' ; you can make both the width and the height anything from 1 to 8 times the size of your real desktop. Don't forget that the desktop has to show on the icon or window! The bigger desktop you have the smaller the detail on the map will be.

BigDesk - Snap Edge

When moving application windows around on the BigDesk map, you may want them to be aligned on the grid (if you have it enabled).The value here defines how close (in pixels on the map display) the top left hand corner of the application window has to be to a grid point for the window to be snapped to that grid point. This option only works if you have Snap Desktop to Grid   enabled. To turn off this feature, give it a distance value of 0. This option is set in the 'Set Options...' dialog box.

BigDesk - Display Window

This option, set from the 'Set Options' dialog box, lets you decide how BigDesk displays windows on it's map. There are three choices:-

Title Bar	displays the window's title along the top of the rectangle.
Application Icon	Displays the icon in the centre of the rectangle.
Both	Does what it says!

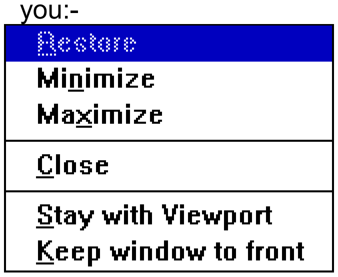
BigDesk - Exit BigDesk

You can decide what BigDesk is to do with all the windows spread over the virtual desktop when you close it. There are 4 options, one of which can be chosen from the 'Set Options...' dialog box:-

Scale To Screen	Move all windows onto the screen, giving them a position which roughly denotes where they were on the virtual desktop.
Cascade All Windows	Move all windows onto the screen offsetting each a little to the right and down from the previous.
Tile All Windows	Move all windows onto screen and arrange them so they cover the screen.
Do Nothing	Leave all windows in their current positions, regardless of where they are on the virtual desktop

BigDesk - The Application Menu

If you ALT+Left Click on any of the applications in the map window, a menu is displayed which lets you do all manner of interesting things to that window. This menu is shown below. Please note that some of the options shown here may not be available (they will be greyed out). Click on an option which interests you:-



Restore/Maximize/Minimize

Change the state of the window (just like the real Restore/Maximize/Minimize from the real system menu). Some of these will be greyed out if the window can't (or doesn't want to) change to a particular state.

Close

Closes the application Please **don't** try and use ALT+F4 as this will close BigDesk instead.

Stay With Viewport

If you choose this option, the window will remain with the viewport regardless of where you move it. You get a window that will follow you around as you move over the virtual desktop (just like the BackMenu attribute TODesk). You can toggle this on or off and the state will be saved if you use the "Save Desktop File..." menu option

Please note this works for windows which are not currently visible as well as those which can be seen. If you give this property to a window which is above the current viewport it will follow you around but stay above the viewport. Useful?

Keep Window To Front

This option allows you to make the window float above all other windows (just like the BackMenu attribute TOFront). You can toggle this on or off and the state will be saved if you use the "Save Desktop File..." menu option

BigDesk - Drag'n'Drop

Do you want to run programs quickly and have them start up in a particular place on the virtual desktop? If typing in a BackMenu command line isn't for you, then the drag'n'drop support in BigDesk certainly is.

You can drag a file from File Manager (or one of your aliases) and drop it on the BigDesk map. The point at where you drop it will be the point at which it will be started up   Please note that if you do drag an alias, BigDesk will ignore any special command line parameters or attributes you may have specified.

BigDesk - Desktop Saving and Loading

Want to try and capture a particular session and come back to it later? The BigDesk desktop loading and saving lets you do that (or at least come close).

To save the current set of applications, choose 'Save Window Positions...' from the 'Desktop' choice on the system menu. A dialog box will appear, asking where you want to save the details; you can choose the default (BIGDESK.DTF) or select a different file.

To load a desktop choose 'Restore Window Positions' from the 'Desktop' choice on the system menu, select a file containing the desktop layout and BigDesk will try and restore it.

What Information Can BigDesk Really Save?

We'd be lying if we said that the desktop saving process captured the entire state of all of the applications you were using and would let you continue from exactly where you left off; that's a whole other can of worms. What BigDesk can do is work out which applications are running and saves a sequence of commands which will restore that layout. It saves the following

- The names of the executable files needed to run each application (along with any command line parameters used when the application was run).
- Position information for each of the windows used within each application.

How does BigDesk restore the desktop

The procedure BigDesk follows when loading a desktop file is simple. For each of the applications BigDesk has to load, it first looks to see if that application is running. If it is, BigDesk repositions it so the window or windows are in the saved positions. If the application cannot be found, BigDesk executes it using the command line information found during saving, and then positions any windows it can identify.

Why might My Desktop not be restored properly?

There are a number of reasons why the desktop might not appear as expected :-

- Any windows which were not around when the desktop was saved will not be moved. This includes any windows which might be created as part of starting an application, but were subsequently closed **before** the desktop was saved.
- Any application which has a dialog box or other window which is appears before the main application window. In this case BigDesk will probably apply the position information to the dialog box and not the main window. To stop this from happening, define a rule in the rule-base for that application with a suitable DELay attribute so BigDesk waits for the dialog box to disappear. This is only a problem if the application isn't running when you load the desktop.
- BigDesk is only able to extract the command line used to start an application. If you didn't run an application with any parameters, or you interactively opened a file, BigDesk will not be able to have the appellation open the correct file for you.

To try and help overcome some of these problems, a rule-base is provided which lets you tailor the way BigDesk saves information about particular applications.

BigDesk - The Rule-Base for Desktop Saving and Loading

The biggest headache when trying to write a general desktop saving/loading routine is catering for all of the special cases. It's possible to write code that'll work for 90% of the time, but you can guarantee that most people run the applications that fall into the other 10%. So, to try and provide the ultimate desktop software, we have introduced the BigDesk rule-base.

The rule-base lets us (and you) cater for that other 10% of applications where some fiddling is required. The way it works is very simple. There is a file called BIGDESK.RUL that contains a list of application file names (or wild card specifications) which can modify how the entry for that application get saved when the locations of the windows on the desktop are saved.

The rule-base is split into two parts, program name mapping and attribute mapping. The rules are applied in the order they are listed in the rule-base. The format for a rule-base file is very simple:-

```
[Program Names]
OldProgam=NewProgram
...

[Program Attributes]
Program=Attributes
...
```

BigDesk - Rule-Base Program Name Mapping

The program name mapping section "`[Program Names]`" allows BigDesk to map from the names of applications found when interrogating the Windows task database, to the names of applications that you might want to save. An example of why you need this is an application called Lotus Organiser. As far as Windows is concerned, the task is called ORGANISER.BIN. If BigDesk were to save that name. it would not be able to re-load the application.

Each entry in the program name mapping takes the form:-

```
OldProgramName=NewProgramName
```

Where OldProgram name can be explicit or contain wild-card characters (* ?).For the organiser problem above, a rule might look like the following:-

```
ORGANISER.BIN=c:\winapps\organise\ORGANISER.EXE
```

and BigDesk will do the translation when it comes to saving the desktop. Note you can have wild-card characters on the left of the =, but not on the right. So you can't have:-

```
*.BIN=*.EXE
```

which we admit is very useful, but we haven't got round to writing it yet!

You can have:-

```
*.BIN=c:\tmp\foo.exe
```

which will match anything ending in .BIN and replace it with c:\tmp\foo.exe

The program name mapping section also lets you define which applications are not saved. If you run BackMenu as you shell, then there's no need to save it as part of the desktop. By having an entry in this list with nothing on the right-hand side, an application can be "ignored". For example:-

```
BACKMENU.EXE=
PROGMAN.EXE=
```

will cause BigDesk to ignore BackMenu and Program Manager when saving the desktop. Note that if you change the name of a program, then it will be the **new name** that is used to search the attribute mapping section, and not the old one. Simple, isn't it!

BigDesk - Rule-Base Program Attribute Mapping

The attribute mapping section "`[Program Attributes]`" lets you fiddle with the attributes that are saved as part of the desktop file. You can do three things

- Override the value of an attribute
- Remove an attribute
- Leave the attribute value alone

Each entry in this section takes the form:-

```
ProgramName=List of Attributes
```

Where the ProgramName can be explicit or contain wild-card characters. You can have rules like:-

```
WINFILE.EXE=NOCMDLINE
*.EXE=NOWIDTH
```

which will mean that BigDesk will not try and save the command line parameters for File Manager and all .EXE programs will not have any width information stored about them. To set an attribute value, you use the attributes as defined by the BackMenu command line syntax For any attribute that takes a value, however, there is now an equivalent that removes that attribute. Most of them are obvious as they have the name NOxxx. Here's a complete list:-

```
NODIR, NOXPOs, NOYPOs, NOXDEsk, NOYDEsk, NOWIDth, NOHEIght,
NODELay, NOSNAp, NOCMDline, NOTOFront, NOTODesk
```

You can also re-enable attributes again (i.e. turn them back on). This can be done one of two ways. Either you can specify an attribute with a particular value, or you can specify an attribute with no value, and the previous value is reinstated. Here's a few examples that might clarify this.

```
[Program Attributes]
*.EXE=NOWIDTH
BOXWORLD.EXE=WIDTH=100
```

BigDesk is going to save the position of BOXWORLD.EXE. It currently exists at (100,150) on the screen and is 200x200. The rules are parsed in the order in which they occur. So, for BOXWORLD.EXE, the first rule we meet is *.EXE. This rule matches and so the width attribute is removed from the settings for BOXWORLD. The next rule that matches is the one for BOXWORLD.EXE itself, and this reinstates the width parameter and sets it to 100. The entry saved for BOXWORLD.EXE would be:-

```
{XPOS=100 YPOS=150 WIDTH=100 HEIGHT=200}BOXWORLD.EXE
```

If, instead, the second rule was replaced with:-

```
BOXWORLD.EXE=WIDTH
```

then the WIDTH attribute would be reinstated and retain its old value. The entry saved this time would be:-

```
{XPOS=100 YPOS=150 WIDTH=200 HEIGHT=200}BOXWORLD.EXE
```

Finally, if the rule for BOXWORLD.EXE is removed entirely, then the entry saved is:-

```
{XPOS=100 YPOS=150 HEIGHT=200}BOXWORLD.EXE
```

A note about DOS based programs

When saving the desktop, BigDesk is able to extract the names of any DOS-based applications running as DOS boxes or full screen applications. It cannot detect, however, what state such an application is in and so may not restore it properly. Each DOS based application uses the rule-base in the same manner as windows programs except it is possible to provide settings that apply to **all** DOS-based programs. Whenever a DOS program is having the rule-base applied, BigDesk also looks for an entry for the Windows component (WINOA386.MOD for Enhanced mode and WINOA286.MOD for standard mode). By having one of these as an entry in the rule-base, attributes can be applied to all DOS-based programs.

For example:-

```
WINOA386.MOD=DELAY=500
```

This will cause the DELay attribute to be set for every and any DOS program saved as a part of the desktop.

Don't save the working directory for any applications matching this rule

Don't save the X co-ordinate of the window for any applications matching this rule

Don't save the Y co-ordinate of the window for any applications matching this rule

Don't save the X co-ordinate of the desktop for any applications matching this rule

Don't save the Y co-ordinate of the desktop for any applications matching this rule

Don't save the width of the window for any applications matching this rule

Don't save the height of the window for any applications matching this rule

Don't save the initial delay for any applications matching this rule (this is only useful if a previous rule has defined a DELay)

Don't save the snap-to-window state for any applications matching this rule (this is only useful if a previous rule has used SNAp)

Don't save the command line parameters for any applications matching this rule, Don't save the to-front state for any applications matching this rule, Don't save the keep-with-desktop state for any applications matching this rule,

BigDesk - Access From Other Applications

The BackDesk DLL (BACKDESK.DLL) contains two accessible functions that can be called from other applications to let you gain access to BigDesk. They are:-

integer **BD_IsBigDeskRunning()**

Is BigDesk installed and running? This function returns an integer which is 1 if BigDesk is running and 0 if not.

ActivateBigDesk()

Allows the BigDesk map window to be given the focus. This function takes no parameters.

Here is a Microsoft Word-For-Windows macro that demonstrate how to use these functions

```
Declare Sub ActivateBigDesk Lib "BACKDESK.DLL"
Declare Function BD_IsBigDeskRunning Lib "BACKDESK.DLL" As Integer

Sub MAIN
    If BD_IsBigDeskRunning <> 0 Then
        ActivateBigDesk
    Else
        MsgBox "BigDesk isn't running"
    End If
End Sub
```

Writing a similar Excel macro is left as an exercise for the reader.

What is WRunServ/WRun?

Ever been running a DOS box within Windows and wanted to start up another Windows application without leaving the command prompt? If you're running Windows in 386 Enhanced mode, then the WRunServ/WRun utilities will let you do this. There are two programs:-

- WRunServ is a Windows application that has to be run to make this magic happen.
- WRun is the DOS program that you use to run other Windows (or even DOS) applications from the command prompt.

For more help, choose an item below:-

What is WRunServ?

What is WRun?

How do I get it all to work?

WRunServ - The Windows WRun Server

WRunServ has to be running in order for WRun to work. Normally, when WRunServ is run, it appears as an icon. It can't do much once started. There's an impressive about box and that's about all. If you don't want the icon cluttering up your desktop, you can run WRunServ with the -hide parameter. No icon, no clutter (well, less clutter), and all of the same functionality.

WRun - The DOS->Windows command line parser

WRun accepts any commands using the BackMenu command syntax, so you can use any of the attributes to control how the application is run. Note that you can't use any of the keywords with WRun (e.g. $SaveAlias).For example:-

```
WRUN notepad.exe c:\autoexec.bat
```

starts up notepad with your autoexec.bat file loaded. Also:-

```
WRUN {XDESK=1 YDESK=1 WIDTH=400 HEIGHT=400}clock.exe
```

will start up the clock application on desktop [1,1] size 400x400.

WRun/WRunServ - How do I get it all to work?

To make this magic happen simply:-

1. From within Windows, start the WRunServ application. If you're going to use WRun a lot, then we suggest you put WRunServ in your Program Manager Start-up group. If you don't want the WRunServ icon cluttering up your desktop, put `-hide` on the command line.
2. From a command prompt, type: `WRUN the-application-you-want-to-run`.

That's it, enjoy.

WCopy and WPaste - access to the Windows clipboard from DOS

WCopy and WPaste are two command prompt utilities which allow you to copy from and paste to the Windows clipboard. Windows must be running in 386 Enhanced mode for these utilities to work. In brief:-

- WCOPY takes stdin and places it on the Windows clipboard as text.
- WPASTE takes the text on the Windows clipboard and copies it to stdout.

Because WCopy and WPaste use stdin & stdout, you can use all of the DOS file redirection operators (> , < , |). For example, to copy your autoexec.bat to the clipboard, type:-

```
WCOPY < c:\autoexec.bat
```

or, to place the directory listing of the current directory onto the clipboard type:-

```
dir | WCOPY
```

To place the contents of the clipboard into a file, say foo.txt, type:-

```
WPASTE > foo.txt
```

and to view the contents of the clipboard using MORE :-

```
WPASTE | more
```

For more information on using redirection within DOS, see your DOS manual.

SP Services Shareware

Software You Want
At Prices You Can Afford!

BackDesk is distributed as Shareware - you should have received it free of charge (or at most have paid for its delivery) so that you can see if you really like it before you pay for it.

BackDesk is not free or public domain, it is not a broken demonstration or Lite version of something else. It is fully function, commercial software that we have chosen to sell by the *try before you buy* method. We trust you to pay for **BackDesk** just like a shopkeeper would trust you to pay for any other package before leaving the store. If you dont want it, please put it back on the shelf!

If you install **BackDesk** and continue to use any of it as part of the way you work for over two weeks, you must either pay for it or stop using it.

More information:

How to register and pay for BackDesk

Site and multiple licenses

Terms and conditions

Register BackDesk NOW!

If you've found that part or all of BackDesk fits in with the way you work, please register and support an under-paid University research fellow ;-) !

For just £25, you can register THE WHOLE SUITE, and receive a nice certificate on recycled paper and a registration number to allow you to enter your name or other personal details on the opening windows and About... boxes.

We also offer low-cost upgrades to existing users.

We don't supply a disk or a manual with your registration (that's one reason why it's cheap!).

Just print out the order form, fill in the details and fax it to SP Services on (01703) 322416 . If you've no fax, post the form to:

SP Services
PO Box 456
Southampton,
United Kingdom
SO17 1LP

You'll have to do this if you don't want to pay by credit card, too, but please note we prefer payment in pounds sterling. If you really must use a company order and expect to be invoiced, we charge £5 per month for the extra work involved in invoicing, issuing statements, sending in the bailiffs etc. We also accept orders by electronic mail.

Upgrade Pricing

For £15, you can swap an earlier registration for just BackMenu or BigDesk (please quote your serial number when upgrading).

For £10, you can upgrade an earlier BackDesk registration (again, please quote your serial number).

If you need a disk, we'll send you the current version of BackDesk on a disk filled with all our other products too **for £5 extra**.   Please ensure you indicate the disk format on the order!

All the documentation is on the disk in this help file and we prefer not to waste paper, so we don't supply a printed manual.   (In response to customer demand, we can also supply a printed version of this help file for an additional £7 or £12 if you want it delivered by air mail).

REGISTRATION APPLICATION FORM

SP Services, PO Box 456, Southampton, SO17 1LP, UK, Fax: +44 1703 322416

Please mark the options that apply to your order. All prices in Sterling; please add £6 if paying with a non-UK cheque or in another currency.

( ) Please register my copy of BackDesk v3 for £25

( ) Please upgrade me from BackMenu/BigDesk for £15

( ) Please upgrade me from BackDesk v2 for £10

( ) - I need a receipt please. ( ) - Please invoice my company (£5 extra/month due)

( ) - I want a laser printed manual delivered surface mail (£7 extra) ( ) - delivered air mail (£12 extra)

( ) - I want a 3½ product disk containing BackDesk and all other SPS products (£5)

( ) - I want a 3½ disk containing your other favourite non-SPS shareware (£5 extra)

I authorise you to charge my credit card with the total sum identified above; I understand that I will receive one of your very nice registration certificates, appropriate registration number(s) and the undying gratitude of Ian Heath in return for this sum (no disk or manual unless ordered).

I got BackDesk from ______________________ and have version 3.10.

My Details ( ) Upgrades: enter serial number

Name ______________________________

Company ______________________________

Address ______________________________

____

______________________________ Include postal code

Phone ______________________ Fax/e-mail ______________________

Payment Type ( ) Cheque payable to SP Services
( ) Visa ( ) Mastercard ( ) American Express

Card number ______________________________

Card expiry date ________________ (include both start & end date for Amex)

I agree to the Terms and Conditions in the Help File

Signature ______________________________

EXACT personalisation details for opening screen:

"Registered to ____"

That's +44 1703 322416 if you're not in Britain...

International Orders

Please **do** register!   It's really easy; you simply send in your credit card details & address, we debit your credit card in pounds sterling, and the credit card company do all the hard stuff about currency conversion.   <u>As a special incentive, there are no extra shipping costs for non-UK orders!</u>   If you really cannot avoid paying in a currency other than pounds sterling via a UK bank, please add the equivalent of £5 sterling to the price you send to cover part of our currency exchange costs.

We apologise if this seems expensive to US customers, but the price is a fair one for Europe and is distorted by exchange rates and market prices to seem expensive in the US.   At least we're not charging you for shipping!

Electronic Mail

Send orders by e-mail to:

Internet:	**sphipps@cix.compulink.co.uk**
CompuServe:	**100016,1625**

You will get an e-mail reply as soon as the message is received, followed by a registration certificate in the post. *Please supply all the same details with your order*.

We can also provide registered users with support by e-mail or in our support conference on Cix (+44 181 390 1244), called SPS.

Site and Multiple Licenses

If you want to register for a network, or for a mega-corporation, please calculate a fee using the following table.

For three or more registrations, the price is...

up to	10	30	50	75	100	200	copies
each	£20	£17	£15	£12	£10	£9	

Over 200 copies: £1500 one-time charge for first address, then £100 per additional address. We only do bulk registrations of the whole BackDesk package, not the parts seperately. Bulk registrations include one copy of the latest version on disk. Wrun and the clipboard tools are available as a separate OEM package - please contact us for details.

For bulk upgrade pricing, please contact SP Services, fax 01703 322416

Creepy Disclaimer **(the small print in big letters)**

Don't forget that there are no warranties associated with this software beyond saying that it will do what it says in this help file (apart from your legal rights, of course). If you manage to mess up your installation and think it has something to do with BackDesk, we're sorry but it only cost £25, didn't it. Now for BIG money we'll give you a warranty! However, *registered users* may e-mail us or fax 01703 322416 and we will do our best to satisfy your requests. All prices are subject to sudden, unexpected change, and all errors are regrettable, human and apologised for. Once you have a license, you can use our software 'like a book' as they say - do what you wish with it as long as only one person can ever use the licensed copy at any one time. We only sell the permission to use, not the software. (We can translate this section into legalese if you wish!)

DISCLAIMER AND AGREEMENT

(You asked for it!)

Users of BackDesk must accept this disclaimer of warranty. If you do not accept this disclaimer, do not use the programs.

THE BACKDESK SOFTWARE SUITE IS SUPPLIED AS IS. THE PUBLISHER DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THE WARRANTIES OF MERCHANTABILITY AND OF FITNESS FOR ANY PURPOSE. THE PUBLISHER ASSUMES NO LIABILITY FOR DAMAGES, DIRECT OR CONSEQUENTIAL, WHICH MAY RESULT FROM THE USE OF BACKDESK, EVEN IF THE PUBLISHER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

THE LICENCE AGREEMENT AND WARRANTY SHALL BE CONSTRUED, INTERPRETED AND GOVERNED BY THE LAWS OF ENGLAND AND WALES. YOU MAY HAVE OTHER RIGHTS WHICH VARY FROM ONE STATE TO ANOTHER.

BackDesk is shareware, and is provided at no charge to users for evaluation. Feel free to share it with your friends and colleagues, but please do not give it away altered or as part of another system. The essence of shareware software is to provide computer users with quality software without high prices, and at the same time to provide incentive for programmers to continue to develop new products.

If you find the BackDesk or any part of it useful, and you are continuing to use it after the trial period of 28 days, you must make the registration. In return you will be given a user name, serial number and registration number to disable the shareware reminders. This information will be good for all future shareware releases of BackDesk v3.x. You will be informed when major releases become available.

A site licence for over 2 copies may be obtained at a reduced cost by contacting SP Services by post or email.

Any person or organisation wanting to distribute BackDesk for profit must first contact SP Services by post or email for authorisation. Cost recovery is not considered for profit.

Copyright for this software remains the property of Clarity Systems Ltd. BackDesk was written by Dr Ian Heath assisted by Nick Beitner who wrote the BackDrop library. James Pharaoh wrote BME.

If you see this prisoner, do not attempt to approach him; he may be dangerous. In particular he may start wibbling about SimCity, MicroCosm, Yes and other topics. Contact the police at once and please, be gentle, he has had a hard time over the past year being nagged to write BackDesk v3.

I can be contacted at:

Internet: pharaoh@dircon.co.uk

CompuServe: 100541,1116

Snail Mail: James Pharaoh, SP Services, PO Box 456, Southampton SO17 1LP

SP Services
PO Box 456
Southampton
SO17 1LP
United Kingdom
Fax +44 1703 322416
E-mail sphipps@cix.compulink.co.uk
CompuServe 100016,1625

Known Problems with BackDesk v3.10

The following problems were known as of 20th January 1995:

Dustbin may freeze when deleting aliases

BigDesk doesn't save File Manager details correctly

If you encounter problems you can't solve, please report by e-mail with full details of the problem and how to recreate it, plus details of your system and its configuration (a listing as produced by MSD.EXE is helpful, as is a Dr Watson trace if available). Note that we cannot fix problems we can't reproduce, nor do we promise to neccessarily make *any* changes.

Dustbin may freeze the system

On certain machines it has been reported that dustbin can freeze the computer when deleting aliases. We don't know whats causing it yet so we thought we'd tell you about it anyway. This should be fixed in a future release.

BigDesk doesn't save File Manager details correctly during desktop save

File Manager, for some reason, causes BigDesk problems when you try and save a desktop layout with it running. Problems include saving garbled characters for the File Manager command line, or not saving the applications file name at all.

Writing External keyword DLLs for BackDesk V3.10

BackDesk now provides a powerful way for programmers to extend the functionality of the system without having to modify the BackDesk application. Instead, you can now write external keyword DLLs, and attach these to BackDesk via the SP-SERV.INI file. Once this has been done, BackMenu will automatically make these new keywords available to the user in a transparent manner. You have complete access to the "command lines" for these keywords and can parse and detect errors.

In order to develop external keywords for BackDesk, youll need the BackDesk external keywords SDK. It contains all of the header files and libraries required for both C and Pascal development. Look for BDSDK3.ZIP the same place you found this package or contact S.P. Services for more details.

What can I do with these keywords?

How do I write an external keyword handler for BackMenu?

What are these functions I have to write?

So how do I use my shiney new external keyword in BackMenu?

Where have all these constants and structure come from?

What can I do with these external keywords?

Anything you want to do! The system is completley flexible and caters for any kind of external keyword. This can range from simple keywords (like $About) which can take arbitrary parameters to more complex keywords generating multiple cascading menus (like $Groups).

How do I write an external keyword handler for BackMenu?

Within a BackMenu menu, each entry has a unique identifier which is used by BackMenu to determine which entry was chosen (Command IDs). As the menu is built, BackMenu allocates a command ID to each description in the menu and keeps track of the command line associated with that description by using the Command IDs. The same is true for the current BackMenu keywords. Each instance of a keyword has its own Command ID (or possibly set of Command Ids) which are used to make the magic happen.

The external keywords work by having each instance allocate itself a set of unique Command IDs from the external keywords Command ID pool. BackDesk then uses these IDs to work out which external keyword was chosen from the menu, and calls the relevant function inside the external keyword handler.

A keyword can occur inside a menu many times. BackDesk allocates each instance a unique ID and so can tell them apart (useful if you've got a keyword which can have different command lines). By default, each keyword (and each instance) will use one Command ID. It is possible, however, to ask for more.

When BackMenu builds the menu, you will be given a chance to insert into that menu via the "parse" function. Whenever BackMenu comes across one of your keywords, it will call your supplied "parse" function, which can then go about the merry business of checking any command line supplied to your keyword, constructing a cascading menu (if you need to) and generally having a good time.

The parse function can be called in one of two circumstances

1. BackMenu is performing a keyword integruity check - so all you have to do is check any command line parameters to your keyword and tell BackMenu whether they are worthy or not.
2. BackMenu is building the menu, so you can do any extra stuff like building a cascading menu to hang off your keyword.

You also have to tell BackMenu at this point how many Command IDs you've used. For a simple external keyword, this will be 1, but for complex cascading menus you'll obviously need more. If the number of Command IDs that you are going to use will change dynamically during the lifetime of your cascading menu, we recommend that you build an upper limit into your DLL and ask for that many. Please also note that there isn't an inexhaustible supply of these Command Ids, so don't ask for 2000 of them when you're only really ever going to use 200 - Its not very friendly :-<

If you do want to build a cascading menu, you might want to modify it before it is displayed to the user (like the $Tasks menu which always shows the current set of Windows open). To do this, you provide a "Popup" function. It will be called just before your menu appears, allowing you to perform any modifcations (except deleting it :->). Anyone who has written code for WM_INITMENUPOPUP will understand whats going on here.

If you do modify your menu, remember the golden rule:-

> Don't use any command IDs outside of the range given to you. (The Popup function is given the base Command ID and number of Command IDs used for this instance of the keyword). If you do use IDs outside your alotted range, nasty things will (probably) happen.

That only leaves one thing - how to make anything actually happen. This is where your "Cmd" function comes in (this is the only function that HAS to be in the DLL. All of the others are optional). Your "Cmd" function will be called if one of two things happen.

1. The user clicks on your keyword (or a part of the cascading menu for your keyword).
2. The executes a $keyword by some other means (part of an alias, typed in the "Execute" dialog box.

Its up to you to do the hard part - making things happen.

All you have to do is create a Windows dynamic link library using you favouriteprogramming language. This DLL has to contain one or more of the functions described below. Once the DLL is written, you can "install" it by modifyingSP-SERV.INI, or using the supplied $XKeywordManager external keyword. Once this is done, the new external keyword is available and can be used in the BackMenu menu or even from another source (e.g. alias, command line...)

What are these functions I have to write for an external keyword?

There are 6 different functions you can provide in your DLL which will be used by BackMenu to make the magic happen! Of these functions, only one is required, all of the others are optional - making it very simple to add keywords. The basic rule is:-

The more complex your Keyword, the more functions you must support.

Here are the functions - in each case the function name is prefixed with a string which is defined for each external keyword. This means you can have handlers for several keywords in the same DLL. In the examples shown below, the prefix is denoted by <KeywordPrefix>.

<KeywordPrefix>Init()

<KeywordPrefix>Parse()

<KeywordPrefix>PopUp()

<KeywordPrefix>Cmd()

<KeywordPrefix>Info()

<KeywordPrefix>Done()

KeywordLibInfo()

<KeywordPrefix>Init()

C	**int**	**<KeywordPrefix>Init(void)**
Pascal	**function**	**<KeywordPrefix>Init: integer; export;**

Description

This function is called once when your DLL is loaded by BackDesk.

Retun Value

<> 0 if your initialisation was sucessful, 0 if not.

Comments

None

<KeywordPrefix>Parse()

C	**int**	**<KeywordPrefix>Parse(LPSTR szCmdLine, WPARAM wCmdIDBase, KEYWORD_PARSE_PTR kppInfo)**
Pascal	**function**	**<KeywordPrefix>Parse(pszCmdLine : pChar; wCmdIDBase : word; pkpInfo : KEYWORD_PARSE_PTR): integer; export;**

Description

Parses an instance of your keyword in the menu

Parameters

szCmdLine	the "command line" for this keyword.
wCmdIDBase	the start of the set of Command IDs allocated for this instance of your keyword.
kppInfo	A structure you fill in to return extra information to BackMenu

Return Value

0 if the parsing is sucessful, < 0 if not. If you return an error <= KEYC_USER_ERROR_BASE, BackMenu will display the user defined error associated with the value (otherwise, a generic error message is displayed).

Comments

This function can be called on two accounts. First is the BackMenu menu integruity check. In this case wCmdIDBase will have the value KEYWORD_CMD_PARSE and kppInfo will be NULL. All you have to do is look at szCmdLine and tell BackMenu whether it is OK or not. The other time this function is called is when the menu is actually being built. In this case, wCmdIDBase is the base command ID for this instance of the keyword, and kppInfo points to a parsing structure which has three fields :-

kppInfo->wFlags	set on input and output with information aboutthe parsing
kppInfo->hMenu	set on output if you build a cascading menu.
kppInfo->wNumCmdIDs	set on input to 1, set this to the number of Command IDs you've used in creating this instance.

Only one flag of wFlags may be set when this function is called, KEYF_IS_INVISIBLE. If this flag is set, your menu occurs inside an invisible menu (one started by <Menu-Name rather than >Menu-Name). If this is the case, DON'T actually build your menu as the value of kppInfo->hMenu will be ignored and you'll use up valuable system resources.

The other flags in kppInfo->wFlags are:-

KEYF_GREYED	which you can set if you want your keyword entry to appear greyed out.
KEYF_MENU	which you set if you've created a cascading menu and put the handle in kppInfo->hMenu.
KEYF_POPUP_SUBMENUS	which lets you receive notification when ANY of the sub-menus in your menu are activated and not just the top-level menu (requires KEYF_MENU)

<KeywordPrefix>PopUp()

C	**BOOL**	**<KeywordPrefix>PopUp (HMENU hMenu, HMENU hSubMenu, WPARAM wCmdIDBase, WPARAM wNumCmdIDs)**
Pascal	**function**	**<KeywordPrefix>PopUp(MenuHandle, subMenuHandle : HMENU; wCmdIDBase : word; wNumCmdIDs : word): boolean; export;**

Description

Called just before a menu, created during <KeywordPrefix>Parse, is displayed.

Parameters

hMenu/MenuHandle	The menu handle of the entire menu created by the keyword
hSubMenu/SubMenuHandle	If the KEYF_POPUP_SUBMENUS flag was specified at parse time, this parameter contains the menu handle of the currently activated sub-menu within your menu. If not, it will contain the same value as hMenu.
wCmdIDBase	the start of the Command IDs for this instance of the keyword
wNumCmdIDs	The number of Command IDs used in this instance.

Returns

TRUE if all goes well, false if there's a problem.

Comments

At this point you can use the SDK menu functions to modify (but not delete) the menu referred to by hMenu. You may modify/delete any sub-menu menu contained within the your keyword's menu hierarchy (e.g. the menu passed in as hSubMenu). Be aware that hSubMenu can be the same menu as hMenu, so exercise caution before deleting hSubMenu. This function is called as a direct result of a WM_INITMENUPOPUP, so use the same rules you'd use in handling that message.

<KeywordPrefix>Cmd()

C	**int**	**<KeywordPrefix>Cmd(WPARAM wCmdID, WPARAM wCmdIDBase, WPARAM wNumCmdIDs, HMENU hMenu, LPSTR szCmdLine)**
Pascal	**function**	**<KeywordPrefix>Cmd(wCmdID : word; wCmdIDBase : word; wNumCmdIDs : word; MenuHandle : HMENU; pszCmdLine : pChar): integer; export;**

Description

Do your stuff - this function allows your keyword to actually perform some task.

Parameters

wCmdID	The ID of the item actually chosen (it may not be wCmdIDBase if you've got a cascading menu).
wCmdIDBase	The base values of Command IDs for this instance of the keyword (an effective way of differentiating between instances of the same keyword).
wNumCmdIDs	The number of Command IDs used in this instance.
hMenu/MenuHandle	The BackMenu menu (so you can get at any menu entries you wish)
szCmdLine	The "command line" for this instance of the keyword. This will be set even if the wCmdID is part of a cascading menu.

Return Value

This function has many return values, all of which are supplied in XKEYBITS.H. An error condition is signified by returning a value < 0. The only errors that should be returned are:-

KEYC_NO_EXECUTE	Returned if this keyword can only exist within a menu (i.e. it cascades) and the user has tried to execute it elsewhere (e.g. Alias).
KEYC_RELOAD_MENU	Not really an error - it tells BackMenu to reload the current menu (useful for certain keywords that change the state of the system e.g. $XKeywordManager).
KEYC_USER_ERROR_BASE	You can return a user defined error value (<= this value). BackMenu will then ask you for a string to describe this error via the <KeywordPrefix>Info function described later.

Comments

This function will be called in one of two circumstances. Firstly, the user might cause the keyword to be executed in some manner other than selecting it from a menu. In this case, wCmdID and wCmdIDBase are both KEYWORD_CMD_EXECUTE, and hMenu is NULL.

Otherwise, the "Cmd" originated from a menu. Its up to you to decide what to do.

<KeywordPrefix>Info()

C	**int**	**<KeywordPrefix>Info(WPARAM wInfo, LPSTR szBuffer, WORD wBufSize)**
Pascal	**function**	**<KeywordPrefix>Info(wInfo : word; szBuffer : pChar; wBufSize : word): integer; export;**

Description

Called by the system to extract information about your keyword.

Parameters

wInfo	The information requested (see Comments).
szBuffer	A buffer into which the information is to be placed.
wBufSize	The maximum size of that buffer.

Return Value

Error is < 0, otherwise return TRUE or an appropriate value depending on the information requested (see Comments).

Comments

The wInfo parameter defines the information wanted. It will be one of the following:-

KEYI_KEYWORD	Return the Keyword string in szBuffer (e.g. "XKeywordManager").
KEYI_DESCRIPTION	Return a concise description of the function of this keyword in szBuffer.
KEYI_PREFIX	Return the Prefix string used to locate the functions within this DLL in szBuffer.
KEYI_ICON_FILENAME	Return the name of the file containing the icon to be used for this keyword in szBuffer.
KEYI_ICON_NUMBER	Return the index of the icon within the file given by KEYI_ICON_FILENAME in the return value of the function. Icon indexes start at 0. Note this is NOT the resource number of the icon within the file.
KEYI_NUM_LIST_DESCRIPTIONS	Return the number of descriptions which can be added to a drop-down list box so this keyword can be used externally (e.g. as an alias). This is returned in the return value of this function. The maximum number of list descriptions allowed is defined by the symbol MAX_LIST_DESCRIPTIONS.
KEYI_LIST_DESCRIPTION	Return a description based from this value, in szBuffer. (e.g. ..DESCRIPTION,.DESCRIPTION+1,..DESCRIPTION+2).
KEYI_USER_ERROR	Return a string describing the user defined error based from this value in szBuffer. This is the absolute value of the error value returned by <KeywordPrefix>Parse or <KeywordPrefix>Cmd.

<KeywordPrefix>Done()

C **void** **<KeywordPrefix>Done(WORD wReason)**

Pascal **procedure** **<KeywordPrefix>Done(wReason : word); export;**

Description

Called when the external keyword is unloaded for any reason.

Parameters

wReason	The reason for unloading. It can be KEYR_TERMINATE (just shutting down), KEYR_NO_FREE_CMDS (not enough Command IDs left) and KEYR_TOO_MANY_CMDS (you've asked for more Command IDs than there are anyway).

Returns

None

Comments

None

KeywordLibInfo()

C	**int**	**KeywordLibInfo(WORD wKeyword, WORD wInfo, LPSTR szBuffer, WORD wBufSize)**
Pascal	**function**	**KeywordLibInfo(wKeyword : word; wInfo : word; szBuffer : pChar; wBufSize : word): integer; export;**

Description

Each external keyword library MUST have this function within it. It only needs to exist once, no matter how many external keywords are supported by the library. It provides a gateway by which the BackDesk external keywords system can interrogate the library and find out its capabilities.

Parameters

wKeyword	The zero based keyword index for which information is required.
wInfo	The information requested (see Comments).
szBuffer	A buffer into which the information is to be placed.
wBufSize	The maximum size of that buffer.

Return Value

Error is < 0, otherwise return TRUE or an appropriate value depending on the information requested (see Comments).

Comments

This funtion mainly acts as a multiplexer. There are two information requests which it has to handle, otherwise it should pass the request on to the relevant keyword information handler. The information values it must support are:-

KEYLI_NUM_KEYWORDS	Return the number of keywords supported by this library as the return vale of this function.
KEYLI_KEYWORD_NUM	A keyword is supplied via the szBuffer parameter and this function must map it to a zero based value which can then be used to refer to the keyword via the wKeyword parameter. This value is returned as the return value of the function or is -1 if the keyword is not supported.

So how do I use my shiney new external keyword in BackMenu?

With the advent of $XKeywordManager, all you have to do is use the keyword to install/remove/edit any external keywords. However, for completness (and in case something goes wrong), this next section is supplied for reference only.

Information about the external keywords is stored in SP-SERV.INI. All you have to do is add the name of your keyword (without the $) and the location of the external keyword library to the list of external keywords which are stored in a section like so:-

```
[External Keywords]
MyKeyword=c:\windows\backdesk\mykeyword.dll
XKeywordManager=c:\windows\backdesk\xkeyman.dll
```

Where have all these constants and structure come from?

xkeybits.h - C include file

```
// Flags for A_KEYWORD_PARSE

#define   KEYF_GREYED           0x0001
#define   KEYF_MENU             0x0002
#define   KEYF_IS_INVISIBLE     0x0004
#define   KEYF_POPUP_SUBMENUS   0x0008

typedef   struct
{
   WORD   wFlags ;
   HMENU  hMenu ;
   WPARAM wNumCmdIDs ;
} A_KEYWORD_PARSE, FAR *KEYWORD_PARSE_PTR ;

// Reason values for _Done

#define   KEYR_TERMINATE        0x0000
#define   KEYR_NO_FREE_CMDS     0x0001
#define   KEYR_TOO_MANY_CMDS    0x0002

// General purpose constants

#define   MAX_KEYWORD_LEN       64
#define   MAX_KEYWORD_CMD_LEN   128

#define   KEYWORD_CMD_BASE      0x8000
#define   KEYWORD_CMD_MAX       0xFFFF
#define   KEYWORD_CMD_EXECUTE   0x0000
#define   KEYWORD_CMD_PARSE     0x0000
#define   KEYWORD_NO_CMD_IDS    0xFFFF

// Keyword CMD return codes

#define   KEYC_NO_MEMORY              -1
#define   KEYC_UNKNOWN_KEYWORD        -2
#define   KEYC_UNKNOWN_INSTANCE       -3
#define   KEYC_NO_CMD_FN              -4
#define   KEYC_NO_EXECUTE             -5
#define   KEYC_RELOAD_MENU            -6
#define   KEYC_NO_KEYWORDS_LOADED     -7
```

```
#define   KEYC_USER_ERROR_BASE  -256

#define   KEYC_OK               1

// Keyword Info constants

#define   KEYI_KEYWORD                1
#define   KEYI_DESCRIPTION            2
#define   KEYI_PREFIX                 3
#define   KEYI_NUM_LIST_DESCRIPTIONS  95
#define   KEYI_LIST_DESCRIPTION       96
#define   MAX_LIST_DESCRIPTIONS       32

// Keyword Library Info constants

#define   KEYLI_NUM_KEYWORDS    128
#define   KEYLI_KEYWORD_NUM     129

#define   KEYI_USER_ERROR       256
```

xkeyword.pas - Pascal TPU source file

```
unit xKeyWord;

{Pascal interface to BackMenu Add-ons API   }
{(c) 1994-5, Clarity Systems Ltd            }

interface

uses winprocs, wintypes;

const
     KEYF_GREYED         =       1;
     KEYF_MENU           =       2;
     KEYF_IS_INVISIBLE   =       4;
     KEYF_POPUP_SUBMENUS =       8;
     KEYR_TERMINATE      =       0;
     KEYR_NO_FREE_CMDS   =       1;
     KEYR_TOO_MANY_CMDS  =       2;
     MAX_KEYWORD_LEN     =       64;
     MAX_KEYWORD_CMD_LEN =       128;
     KEYWORD_CMD_BASE    =       $8000;
     KEYWORD_CMD_MAX     =       $FFFF;
     KEYWORD_CMD_EXECUTE =       0;
     KEYWORD_CMD_PARSE   =       0;
     KEYWORD_NO_CMD_IDS  =       $FFFF;
     KEYC_NO_MEMORY      =       -1;
     KEYC_UNKNOWN_KEYWORD =      -2;
     KEYC_UNKNOWN_INSTANCE =     -3;
     KEYC_NO_CMD_FN      =       -4;
     KEYC_NO_EXECUTE     =       -5;
     KEYC_RELOAD_MENU    =       -6;
     KEYC_NO_KEYWORDS_LOADED =   -7;
     KEYC_USER_ERROR_BASE =      -256;

     KEYC_OK             =       1;

{Keyword Info constants}

     KEYI_KEYWORD        =       1;
     KEYI_DESCRIPTION    =       2;
     KEYI_PREFIX         =       3;
     KEYI_ICON_FILENAME   =     4;
     KEYI_ICON_NUMBER    =       5;
     KEYI_NUM_LIST_DESCRIPTIONS  =      95;
     KEYI_LIST_DESCRIPTION       =      96;
     MAX_LIST_DESCRIPTIONS       =      32;

{Keyword Library Info constants}

     KEYLI_NUM_KEYWORDS   =       128;
     KEYLI_KEYWORD_NUM   =       129;

     KEYI_USER_ERROR     =       256;

type
     A_KEYWORD_PARSE    =       record
                                      wFlags      : word;
                                      MenuHandle  : HMENU;
                                      wNumCmdIDs  : word
                                end; {record A_KEYWORD_PARSE}
     KEYWORD_PARSE_PTR  =       ^A_KEYWORD_PARSE;

{Useful functions from backDesk...                                 }
function BM_BreakMenu(MenuHandle : HMENU): HMENU;
{Splits the menu concerned into balanced columns according to BackMenu
settings  }
```

```
function BM_CopyMax(pszDest : pChar; pszSource : pChar; wcbMax : word):
word;
{Copies as much of Source into Dest as will fit}

implementation

function BM_BreakMenu; external 'BACKDESK.DLL';
function BM_CopyMax; external 'BACKDESK.DLL';

begin
     {No initialisation necessary}
end.
```

Networking - External Keywords

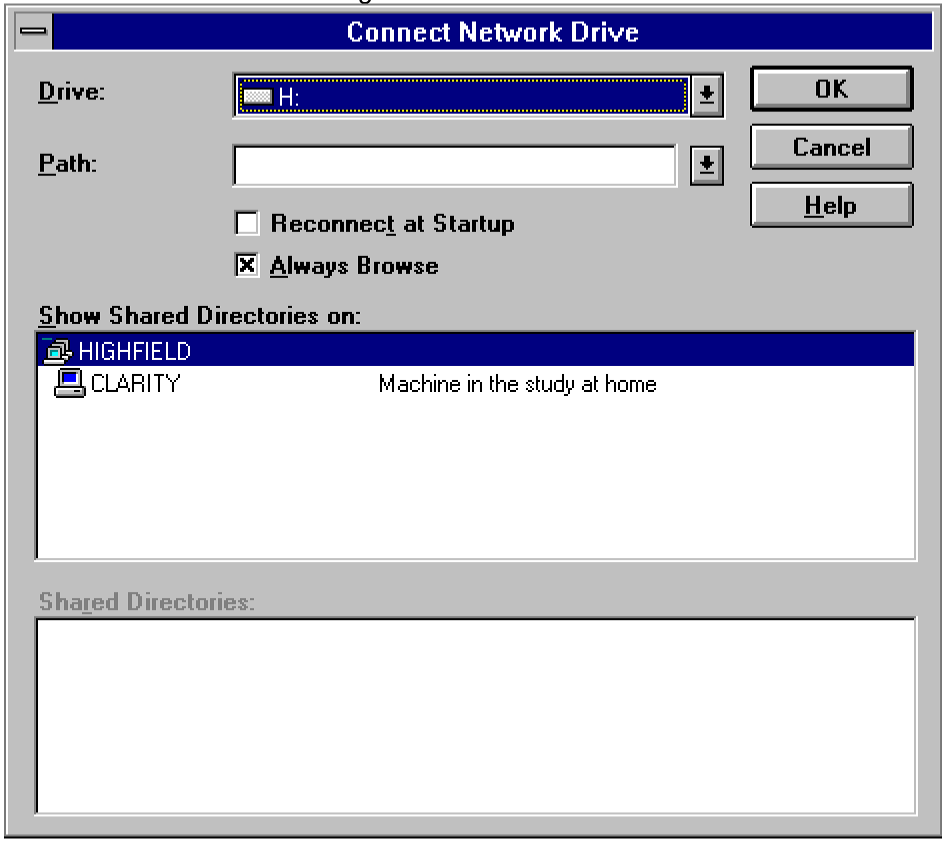
The supplied keyword library, XK_NET.DLL, contains three keywords to help with accessing the network functions in Windows for Workgroups.

$NetConn	Displays a dialog to allow you to add a new network link.
$NetDisc	Displays a dialog showing your current network links and lets you break one.
$NetLinks	Displays a cascading menu of network links that allows you to browse a linked resource and add/remove links.

All of these keywords provide control of both printers and disks. You can add them to your BackMenu installation using the $XKeywordManager command.

$NetConn - Connect a new net link

The **$NetConn** keyword, provided by the XK_NET.DLL external keyword library, displays the standard Windows for Workgroups *Connect Network Drive* dialog. Add the optional parameter Printer to display a *Connect Network Printer* dialog instead.



Select Drive Combo - Use this control to pick the drive letter with which you want to associate this new network link.

Select Network Resource Combo - Use this control to type in a new network UNC name for the resource you want to link or to select one you have chosen previously.

Workgroup System View - The workgroups on your network are shown here. Double-click on a workgroup name to display the machines available in that workgroup. Click on a machine name to view the shareable resources on that machine.

Shareable Resource List - This box displays the resources available for sharing on the system you have selected. Click on a resource to place its name in the Path combo box above.

Add to Reconnection List - Check this box if you want Windows for Workgroups to attempt to link to this resource every time you restart Windows.

Browse Behaviour - Uncheck this box to stop the network search that takes place each time the dialog is displayed.

$NetDisc - Disconnect Net Resource

The **$NetDisc** keyword, provided by the XK_NET.DLL external keyword library, displays the standard Windows for Workgroups *Disconnect Network Drive* dialog. Add the optional parameter Printer to display a *Disconnect Network Printer* dialog instead.

If no resources are currently linked, a message is displayed.

$NetLinks - Cascading Networking Menu

Add **$NetLinks** to your BackMenu to display a cascading menu showing currently linked network drives and printers and offering the opportunity to disconnect or browse them or to connect new ones.

Add the parameter **NoConfirm** to suppress display of confirmation dialogs when resources are disconnected.

Add the parameter **Force** to allow resources that are currently in use to be disconnected.

This keyword is provided by the XK_NET.DLL external keyword library.

BackDesk Command Executor, BDX.EXE

So youd like to use BigDesk but not BackMenu? How do you pass BackDesk parameter lists to BigDesk when executing programs, so that programs appear on the right desktops?

Well, all you have to do is prefix the command line with BDX (assuming BDX.EXE has been placed on your path). BDX will then pass the full command line to BackDesk for execution just like a BackMenu command line. Any valid menu command may be executed, including keywords.

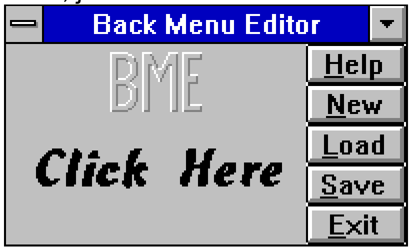
For example,

```
      BDX {TODESK} CLOCK
```

will launch the clock and make it stick to the desktop.

BME - The BackMenu Menu Editor

Welcome to BME, an easy-to-use menu editor for BackDesk. Help can be found on any aspect of using BME, just click on an item of interest from the picture below



You can also get more information on

- Creating a Menu Item
- Editing a Menu Item
- Moving a Menu Item
- Deleting a Menu Item

Help Button

Brings you right here, there's little more to say than that.

New Button

If you're sure you really want to, you can erase all of the entries in the menu being edited. Don't worry, this won't have any effect on the BackMenu menu until you decide to save your work.

Load Button

Displays a dialog box asking for the name of a menu file to load. Choosing one and pressing OK will let you edit the menu contained within that file.

Save Button

Displays a dialog box asking for the file name and location in which to save the edited menu. Please note, if you save your menu to a different file from the one it was loaded, BackMenu will not see that the menu has changed and so the new menu will not be loaded.   Either copy the changes back over the original menu file or change the name of the menu file which BackMenu loads (see $SetOptions for more details)

Exit Button

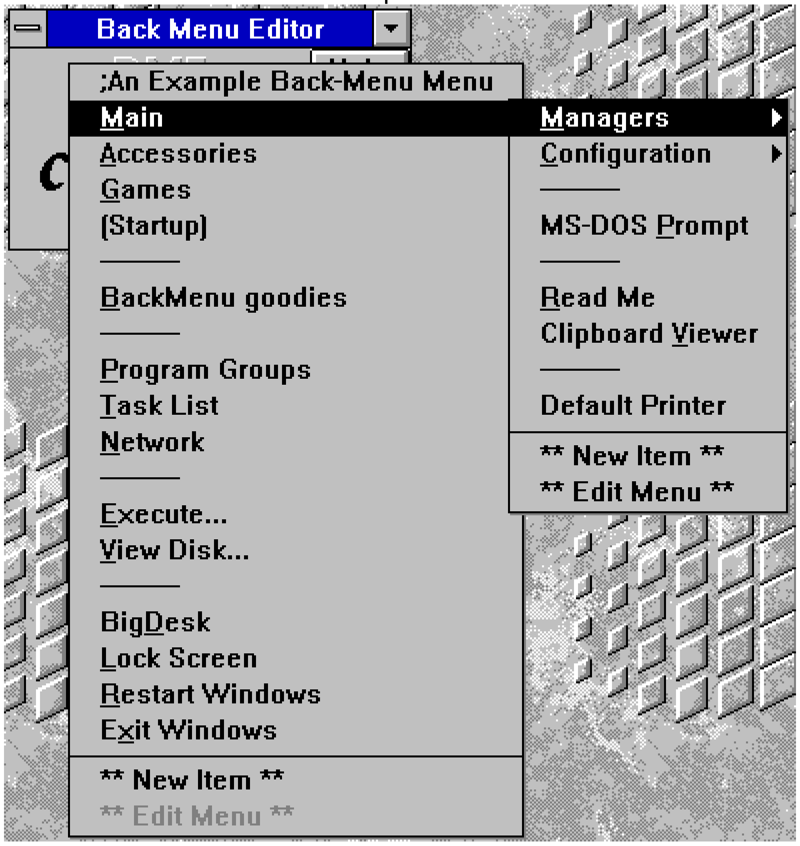
Quits BME and, if you've done any editing, asks if you want to save your shiny new menu.

BME - Editing a Menu

Clicking in the BME window displays the menu you are editing, as shown below. From here you can :-

- Add a new item to the menu
- Edit/Move/Delete an existing item
- Edit/Move the title of a group of items which exist as a pull-right menu

Click on the list above *or* on the picture below for more information



What Kind of Item?

Depending on the type of item chosen, a different dialog will appear to let you edit that item. BME understands about three different types of items:-

- A Menu Entry
- A Comment
- A Separator

Choose the item you are interested in.

A Menu Entry

Selecting this entry will allow you to edit, move and/or delete it. For more information see BME - Editing a Menu Entry.

A Menu Separator

Selecting this separator allows you to move or delete it. For more information see BME - Editing a Menu Separator.

A Menu Comment

Selecting this comment allows you to edit, move and/or delete it. For more information see BME - Editing a Menu Comment.

A Pull-Right Menu

Selecting this menu will cause it's contents to be displayed. You can then edit any of the items which form a part of that menu, or choose to look at any pull-right menus it contains.

New Item

Selecting this item allows you to append a new item to the end of your menu. Once appended, you can edit it like any other menu item. For more information see BME - Adding a New Menu Item.

Edit Menu

Selecting this item allows you to edit and/or move the pull-right menu which contains it. For more information see BME - Editing a Pull-Right Menu.

Edit Menu

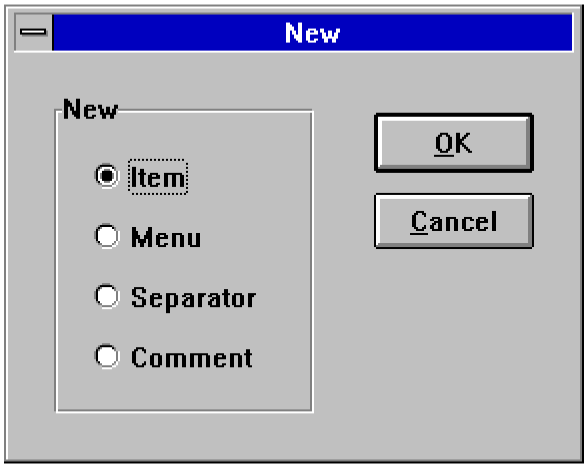
Selecting this item allows you to edit and/or move the pull-right menu which contains it. In the case of the top-level menu, this item is grayed out. For more information see BME - Editing a Pull-Right Menu.

Invisible Pull-Right Menu

Selecting this menu will cause it's contents to be displayed. You can then edit any of the items which form a part of that menu, or choose to look at any pull-right menus it contains. This menu will not be displayed during normal use. For more information see BME - Editing a Pull-Right Menu.

BME - Adding a New Menu Item

By choosing ** **New Item** ** from the bottom of any menu, you can choose to append any one of four different items to the menu as shown below.



Click on an item to find out how to add it.

Item

Appends a new entry to the menu. This is a standard BackMenu entry with a description and associated command to be run. Choosing this will cause a dialog to be displayed which will allow you to create the new entry. For more information see BME - Editing a Menu Entry.

Menu

Appends a new pull-right menu to the menu, displaying a dialog which allows you to create it. For more information see BME - Editing a Pull-Right Menu.

Separator

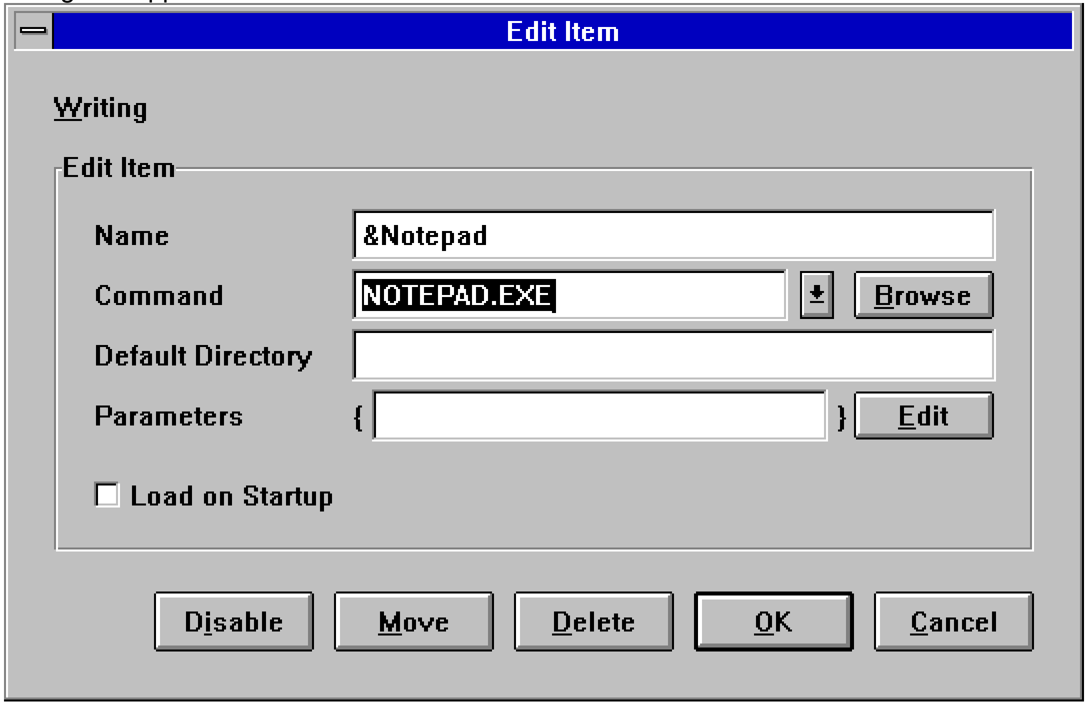
Appends a new separator to the menu. For more information see BME - Editing a Separator.

Comment

Appends a new comment to the menu, displaying a dialog box which allows you to create it. For more details see BME - Editing a Comment.

BME - Editing a Menu Entry

Whether you are editing an existing BackMenu entry or creating a new entry from scratch, the following dialog box appears. Click on an area of interest.



Name: &Notepad
Command: NOTEPAD.EXE
Default Directory:
Parameters: { }
☐ Load on Startup

Name

This is the description of the entry that will appear in the menu.

Command

The command line to be executed. Clicking on the drop down button gives a list of the available keywords (see KeywordCommands for more details).

Browse

Click on this button to display a directory listing box where you can roam your disk looking for possible programs to run.

Default Directory

Enter the name of the directory you want to be in when the program is run.

Parameters

Enter any of the BackMenu attributes here. You can also use the edit button to do this interactively. For more details see BME - Editing a Menu Entry - Editing Attributes or BackDesk Command Line Attributes

Edit

Displays a dialog box where you can edit the individual attributes. For more details see BME - Editing a Menu Entry - Editing Attributes.

Load on Startup

Check this box if you want this command to be run when BackMenu first loads.

Disable

Press this button to disable the entry (comment it out). Useful if youre temporarily removing a program.

Move

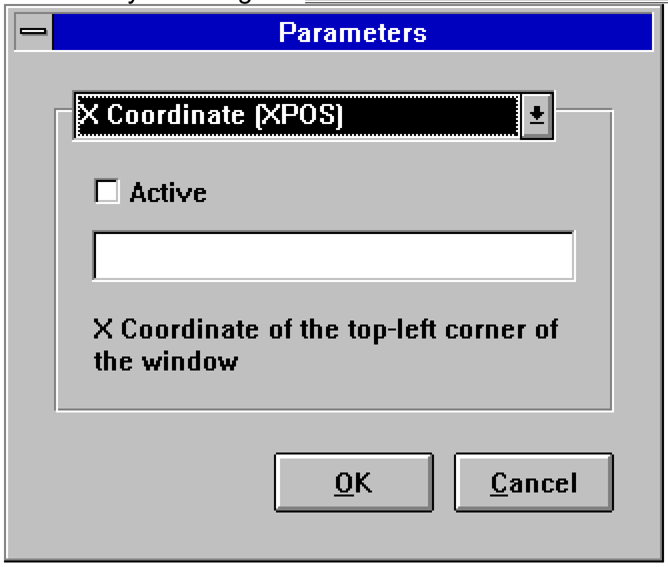
Press this button if you want to move the entry elsewhere within the current menu. For more details see BME - Moving a Menu Item.

Delete

Press here to delete the entry from the current menu.

BME - Editing a Menu Entry - Editing Attributes

This dialog box lets you edit each attribute individually. you can find more out by clicking on the picture below or by checking out BackDesk Command Line Attributes.



X Coordinate of the top-left corner of the window

Attribute List

The complete list of BackDesk attributes is available here. Click on the down arrow to choose the one you wish to edit/include

Active

Check this box if you want to add attribute displayed above to the set for the current entry. Un-check it to remove the attribute.

Parameters

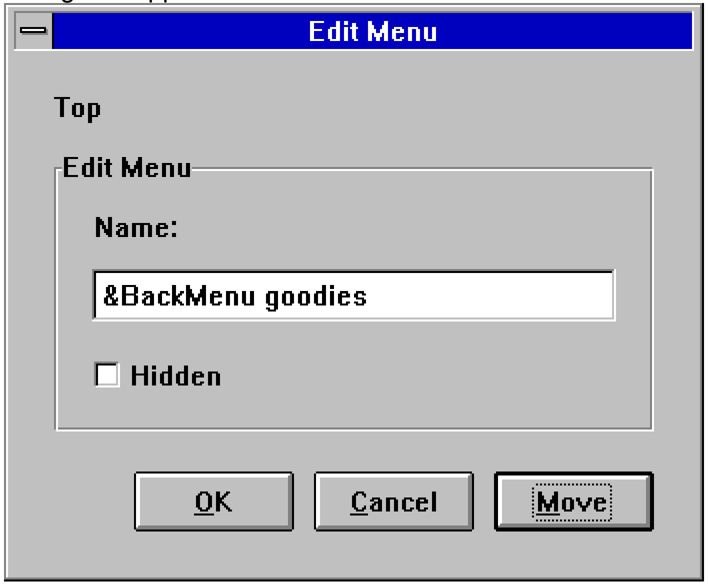
Type any additional parameters required for the attribute here (e.g. the x-coordinate for XPOS). Note that some attributes do not require any additional parameters and so in some instances you will not be able to type here.

Description

Gives a brief description of the chosen attribute

BME - Editing a Pull-Right Menu

Whether you are editing an existing BackMenu menu or creating a new menu from scratch, the following dialog box appears. Click on an area of interest.



Name

Type here the name as you wish it to appear in the parent menu.

Hidden

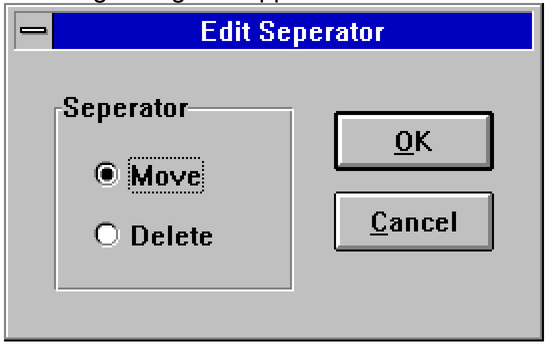
Check this box if you want the menu to be hidden (i.e. not selectable in normal use). Useful if you want to hide away a set of programs to be launched when BackMenu is first started (see the Active check box in BME - Editing a BackMenu Entry).

Move

Press this button if you want to move the menu elsewhere within the parent menu. For more details see BME - Moving a Menu Item.

BME - Editing a Menu Separator

Whether you are editing an existing BackMenu separator or creating a new one from scratch, the following dialog box appears. Click on an area of interest.



Move

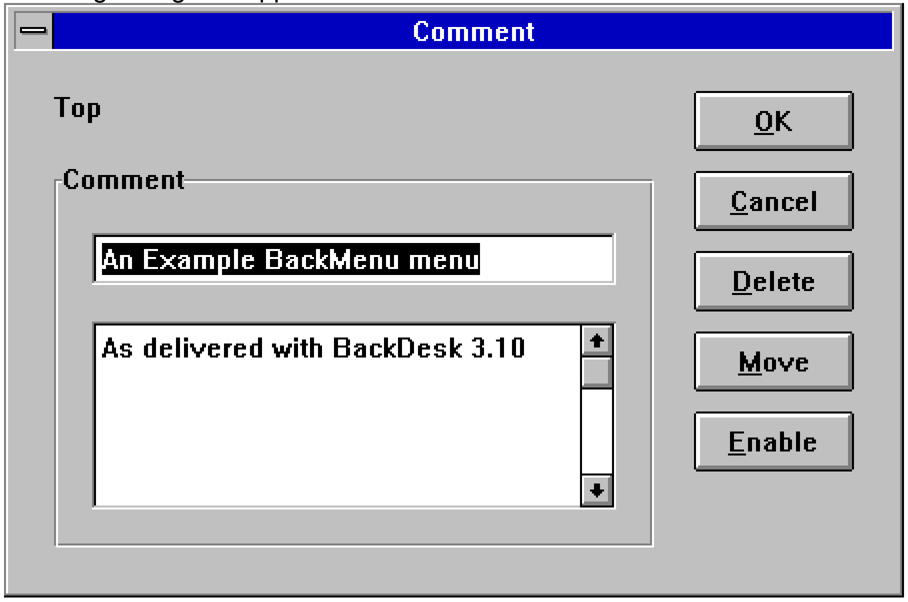
Select this option and choose OK if you want to move the separator elsewhere within the current menu. For more details see BME - Moving a Menu Item.

Delete

Select this option and choose OK to delete the entry from the current menu.

BME - Editing a Menu Comment

Whether you are editing an existing BackMenu comment or creating a new one from scratch, the following dialog box appears. Click on an area of interest.



An Example BackMenu menu

As delivered with BackDesk 3.10

Comment

Enter your comment here. You don't need to place a semi-colon at the start, BME will do that for you.

Extra Stuff

Type an extra information you want associated with the comment

Delete

Press this button to delete the comment from the menu

Enable

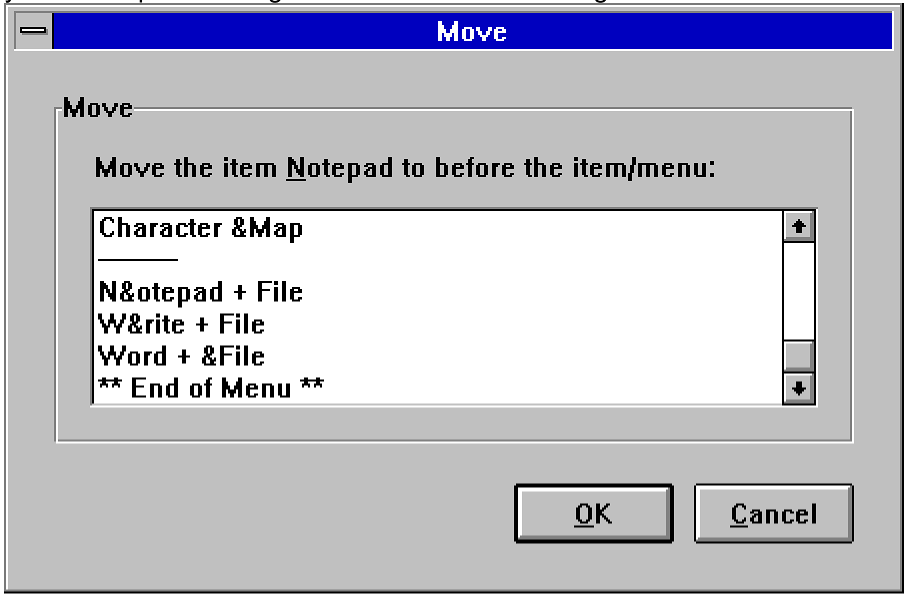
Press this button to enable (i.e. uncomment) a menu entry that you've disabled. For more details see the Disable button in BME - Editing a BackMenu Entry

Move

Press this button if you want to move the comment elsewhere within the current menu. For more details see BME - Moving a Menu Item.

BME - Moving a Menu Item

Whether it is a menu entry, pull-right menu, separator or comment, if you choose to move a menu item, you'll end up at a dialog box which looks something like the one below.



Description

This line will tell you the name of the item you are moving

List

This list shows the names of all of the items contained in the menu within which you are moving your item. Choose the item you wish your item to go before, and press OK.

BME - Deleting a Menu Item

Deleting a menu entry, separator or comment couldn't be easier. Simply press the **Delete button** from within the relevant dialog box. To delete a pull-right menu, simply delete all of the items contained within that menu.