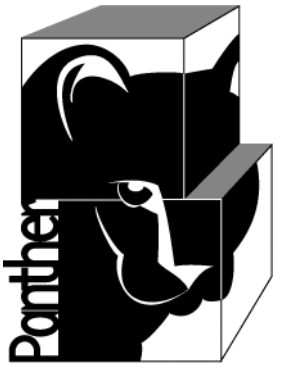
Contents:



Panther

Configuration Guide

Prolifics

Release 5.51

Document 0404
March 2017

Copyright

This software manual is documentation for Panther® 5.51. It is as accurate as possible at this time; however, both this manual and Panther itself are subject to revision.

Send suggestions and comments regarding this document to:

Technical Publications Manager
Prolifics, Inc.
24025 Park Sorrento, Suite 405
Calabasas, CA 91302

http://prolifics.com
support@prolifics.com
(800) 458-3313

Contents:

3. Windows Initialization File

4. Setting Motif Defaults

5. Character Mode Settings

6. Key Translation File

7. Video File

8. Setup File Utilities

Index

About This Document

The *Configuration Guide* provides instructions on how to modify, create, and compile configuration files to customize Panther for your specific requirements. Most of these files are found in the configuration (`config`) directory.

This guide is organized into the following chapters:

- Application variables (containing definitions of setup and behavior variables)
- Windows initialization file
- Motif resource file
- Key file
- Video file
- Character mode settings
- Setup file utilities

What You Need to Know

You should read this guide if:

- You just installed Panther and want more information about configuring Panther—that is, ascertain the location of the required files and/or directories that are installed and used with Panther.
- You want to set new standards or reset defaults for the behavior of your Panther application. All setup options are described in this guide.

Documentation Website

The Panther documentation website includes manuals in HTML and PDF formats and the Java API documentation in Javadoc format. The website enables you to search the HTML files for both the manuals and the Java API.

Panther product documentation is available on the Prolifics corporate website at http://docs.prolifics.com/panther/.

How to Print the Document

You can print a copy of this document from a web browser, one file at a time, by using the File→Print option on your web browser.

A PDF version of this document is available from the Panther library page of the documentation website. You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format.

If you do not have the Adobe Acrobat Reader, you can get it for free from the Adobe website at https://get.adobe.com/reader/otherversions/.

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
Ctrl+Tab	Indicates that you must press two or more keys simultaneously. Initial capitalization indicates a physical key.
italics	Indicates emphasis or book titles.
UPPERCASE TEXT	Indicates Panther logical keys. *Example*: XMIT
boldface text	Indicates terms defined in the glossary.

Convention	Item

`monospace text`	Indicates code samples, commands and their options, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <smdefs.h>`
	`chmod u+w *`
	`/usr/prolifics`
	`prolifics.ini`

`monospace italic text`	Identifies variables in code representing the information you supply. *Example*: `String` *`expr`*

`MONOSPACE UPPERCASE TEXT`	Indicates environment variables, logical operators, SQL keywords, mnemonics, or Panther constants.
	Examples:
	`CLASSPATH`
	`OR`

`{ }`	Indicates a set of choices in a syntax line. One of the items should be selected. The braces themselves should never be typed.
`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed. *Example*: `formlib [-v]` *`library-name`* `[`*`file-list`*`]...`
`...`	Indicates one of the following in a command line: ■ That an argument can be repeated several times in a command line ■ That the statement omits additional optional arguments ■ That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `formlib [-v]` *`library-name`* `[`*`file-list`*`]...`

Convention	Item
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

Contact Us!

Your feedback on the Panther documentation is important to us. Send us e-mail at support@prolifics.com if you have questions or comments. In your e-mail message, please indicate that you are using the documentation for Panther 5.50.

If you have any questions about this version of Panther, or if you have problems installing and running Panther, contact Customer Support via:

- Email at support@prolifics.com
- Prolifics website at http://profapps.prolifics.com

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address and phone number
- Your company name and company address
- Your machine type
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

CHAPTER

1 Overview

Panther applications depend on a number of environment and setup files in order to run. The installation provides the necessary files and, wherever possible, sets default values for required variables. The files and settings that are required vary on each supported platform (Windows and UNIX). Also, two-tier processing and three-tier processing have different configuration requirements. For example, three-tier processing for JetNet or Oracle Tuxedo applications uses server environment files; two-tier processing omits these files.

This chapter describes the different setup files that are provided in a Panther distribution and their relationship to each other.

Environment Setup Files

A Panther executable—whether it runs on a stand-alone workstation, a client workstation, or on a server—relies on correct environment settings. The distribution provides these in the following setup files:

Environment	
Windows client	initialization files: `prol5w32/64.ini, mbedit32/64.ini`
	additional files for JetNet and Oracle Tuxedo applications: `jetman32.ini, jifedt32.ini`

Environment	
UNIX workstation	resource file: `Prolifics` setup file: `setup.sh`
	additional files for JetNet and Oracle Tuxedo applications:resource files: `Jetman, XJIFedit`
	additional files for Panther/WebSphere applications:`prol5unix.ini`
Application server	JetNet and Oracle Tuxedo applications: server environment files—`proserv.env` and `machine.env`*
	Panther/WebSphere applications: `panther.ini` file or environment settings
Web application broker	Web application initialization file (*AppName*.`ini`)

*Supplements settings already established in the environment. (See Chapter 2, "Setting the Enterprise Environment," in the *JetNet Guide/ Oracle Tuxedo Guide*.)

Workstation Configuration Files

The following table describes the files that configure a workstation for Panther, both for development and deployed applications. All files must be in binary format; source is provided in the distribution and can be edited with any text editor. After making changes, you must convert the modified file to binary with the appropriate utility.

Configuration file	Purpose	Conversion utility
`SMVARS` file	Used to find other configuration files and setup information.	`var2bin`

Configuration file	Purpose	Conversion utility
SMSETUP file	Contains application variable settings that supplement or override SMVARS file settings.	var2bin
key translation file	Contains a key translation table for a given terminal.	key2bin
video file	(character-mode only) Supports a given terminal type.	vid2bin

See Also

For information about configuring components of a Panther application, turn to these discussions:

- How to set up a development workstation: see Chapter 6, “Preparing the Development Clients,” in the *Application Development Guide*.
- How to set up the Panther web application broker: see Chapter 2, “Web Application Setup,” in the *Web Development Guide*.
- How to deploy a Panther application: see Chapter 43, “Preparing Applications for Release,” in the *Application Development Guide*.
- How to set up a Panther/WebSphere application server and clients: see Chapter 2, “Configuring Machines,” in the *Panther for IBM WebSphere Developer's Studio*.
- How to deploy COM components in a Panther COM/MTS application: see Chapter 5, “Deploying COM Components,” in the *COM/MTS Guide*.
- How to set up JetNet /Oracle Tuxedo application servers and clients: see Chapter 2, “Setting the Enterprise Environment,” in the *JetNet*/Oracle *Tuxedo Guide*.

CHAPTER

2 Application Variables

Panther application variables can be divided into two general categories:

- *Setup variables* describe the operating environment and point to the Panther installation and specific setup files, and other files required by the application such as libraries.
- *Behavior variables* control the behavior of a Panther application and how users interact with it—for example, message display, cursor behavior, and numeric format.

Unless otherwise indicated, the variables described in this chapter are valid for Panther applications running in all operating environments. Panther has other variables that are applicable to specific operating environments, such as web applications, and character-mode or GUI platforms such as Windows or Motif. These variables are described later in this manual, in one of the chapters that pertain to the supported environments.

The first section of this chapter shows how to set application variables. Variable descriptions follow, organized according to category.

Setting Application Variables

Application variables are set at application startup and affect the entire application. Most are internally set. You can either accept the default settings or explicitly set them yourself in any of these locations, listed in descending order of precedence:

1. The system's environment.
2. Windows initialization file or Motif resources file.
3. The setup file named by the SMSETUP variable.
4. The setup file named by the SMVARS variable. (If SMVARS is not set, Panther expects to find smvars.bin in the config subdirectory.)

Thus, if the application variable SMEDITOR is set in the SMSETUP file and the Windows initialization file, Panther uses the initialization file setting.

Setup Files

You can create or edit a setup file with any text editor. The Panther distribution contains the smvars setup file in its config directory, which you can copy and modify. After you modify setup files, you must convert them to binary format with the var2bin utility.

After you create a setup file and convert it to binary format, set the SMVARS variable to the name of the binary file. If you supplement the SMVARS file with another setup file, you should set the SMSETUP variable to the name of that file.

SMSETUP settings supersede SMVARS settings. Use the SMSETUP file for variable settings that are specific to a particular terminal-type, project, or user. SMVARS typically contains application-wide settings that should be true for all users.

Depending on your platform, you can set an application's SMVARS variable in its environment and in its Windows initialization file. SMSETUP is typically set in the SMVARS file.

Setup File Syntax

Setup file entries use this format:

variable = *value*

value can be a string or keyword, depending on the variable's type (string or integer). For example:

```
TOOLBAR_DISPLAY = TOOLBAR_OFF
```

To continue an entry across multiple lines, end each unfinished line with a backslash (\). To comment out an entry, start it with a pound character (#).

If an application supports multiple terminal types, you might need to qualify some variable settings according to specific terminal types with this format:

`variable` = (`term` [| `term`]...) `value`

`variable` is set to `value` only for those terminals listed within parentheses; `term` is a terminal-type mnemonic as defined by the application variable `SMTERM`. For example, the distributed `SMVARS` file contains these `SMKEY` settings:

```
SMKEY = (ibm)   $SMBASE/config/ibmkeys.bin

SMKEY = (hp|hp2392) $SMBASE/config/hpkeys.bin

SMKEY = $SMBASE/config/xtermkeys.bin
```

Thus, if `SMTERM` is set to `ibm`, Panther reads the first entry and sets `SMKEY` accordingly.

You can provide, along with a number of terminal-specific entries of the same type (for example, `SMKEY` files), one entry that is not terminal-qualified. This serves as the default and it must be last in the list.

Warning: Use terminal mnemonics only if the application runs on different terminal types.

Display Attribute Settings

Some behavior variables take display attributes as parameters. To define a display attribute, select one color and any other attributes through the keywords in Table 2-1. In a setup file, separate the OR'd attributes with blanks, commas, or semicolons, as in this example:

```
STEXTATT = REVERSE HILIGHT GREEN
```

Table 2-1 Display attribute keywords

Foreground Color	Background Color	Attribute
	B_HILIGHT	NORMAL_ATTR
BLACK	B_BLACK	BLANK
BLUE	B_BLUE	REVERSE
GREEN	B_GREEN	UNDERLN
CYAN	B_CYAN	BLINK
RED	B_RED	HILIGHT
MAGENTA	B_MAGENTA	DIM
YELLOW	B_YELLOW	
WHITE	B_WHITE	
	B_CONTAINER	

Portable Settings

When you install Panther, the installation program asks for the base directory for your installation—for example, /usr/panther. This path is used to set the application variable SMBASE. Other variables that point to files in the Panther installation can then use $SMBASE in their file path settings. For example, given the earlier definition of SMBASE, $SMBASE/config/sunkeys.bin expands to /usr/panther/config/sunkeys.bin. If the installation moves, you can update all file pointer settings by changing the value of SMBASE alone.

Runtime Changes

In many cases, you can change application behavior at runtime by setting the appropriate variable with sm_option (for integer variables) or sm_soption (for string variables). For example, this JPL statement disables display of tool bars:

```
call sm_option (TOOLBAR_DISPLAY, TOOLBAR_OFF)
```

To ascertain an integer variable's current setting, call sm_option and supply an argument of NOCHANGE. For example, this statement returns the current value of TOOLBAR_DISPLAY—either TOOLBAR_ON or TOOLBAR_OFF:

```
retval = sm_option (TOOLBAR_DISPLAY, NOCHANGE);
```

To change variables that are set to display attributes, call sm_option and use vertical bars to OR attributes together. For example:

```
call sm_option (STEXTATT, REVERSE | HILIGHT | GREEN)
```

Setup Variables

SMBASE

Points to the root of your Panther installation. In the web initialization file, this variable points to the location of the web application server installation on the HTTP server. SMBASE is set in one or more of the following locations:

- For a three-tier JetNet/Oracle Tuxedo application, in the server environment file machine.env.
- On a client, in the environment or in a Windows initialization file.
- For a web application, in its web initialization file (refer to "Setup Variables," on page 12-1 in the *Web Development Guide*).

For example, you might set SMBASE in a client's UNIX environment as follows:

```
export SMBASE=/usr/panther
```

Similarly, a Windows initialization file can define SMBASE as follows:

```
SMBASE=o:\win32\panther
```

Setting SMBASE facilitates an application's portability across different installations; subsequent definitions of all other variables that point to files can be relative to SMBASE. For example, given the previous SMBASE definition, a client's initialization file can set the SMVARS file as follows:

```
SMVARS=$SMBASE\config\smvars.bin
```

SMCOLMAP

Points to the binary `cmap` file. The `cmap` file defines the default colors, or scheme, for your application, font, color and line style aliases, and the Editor color scheme. You can include a terminal mnemonic in the entry that matches the terminal type designated in your `SMTERM` variable or system `TERM` variable. In this way you can assign color schemes that will take advantage of GUI-specific colors. Refer to "Configuration Map File" on page 45-25 in *Application Development Guide* for more information on creating and modifying configuration map files.

SMDICNAME

(Development only) Identifies the development repository to open when a Panther editing session begins. Only one repository can be open at a time. Panther automatically opens a repository with the name `data.dic`. Specify the repository name. For example:

```
SMDICNAME = /usr/app/dev.dic
```

To specify a remote repository, include the host name with this format:

`host!repositoryName`

For example:

```
SMDICNAME = aspen!myapp.dic
```

SMEDITOR

Specifies the name of the text editor that can be invoked to edit JPL instead of the JPL edit window. If you do not indicate a value for `SMEDITOR`, Panther' text editor is available. For example, your entry might look like:

```
SMEDITOR = vi
```

Use terminal-specific syntax to run the desired editor on a given platform. If you run Panther on Motif, spawn a new window for your editor; otherwise, it runs in the same window used to invoke Panther:

```
SMEDITOR = (xterm) "xterm -e vi"

SMEDITOR = (mswin) notepad.exe %s

SMEDITOR = (ibm) edit
```

You can define `SMEDITOR` in a setup file or the environment, and you can change it at runtime with the library routine `sm_soption`.

SMFEXTENSION

(Backwards compatibility only) Since previous versions of Panther used a default screen extension of jam, set this variable in order to automatically append the specified file extension to any screen name that does not already contain an extension. By default, a period separates the filename from the extension.

```
SMFEXTENSION = scr
```

SMFLIBS

Identifies the libraries to open on startup of Panther or your Panther application in three-tier processing (or in two-tier processing using remote library access). On UNIX you can set SMFLIBS in smvars, smsetup, or in the environment. Under Windows you must set it in the initialization file.
In three-tier applications, set this variable in the server environment file (proserv.env) to ensure that server-specific libraries are open.
To specify multiple libraries, separate their names with a vertical bar (|) or colon, a semicolon under Windows, or the convention used by your operating system. The following examples describe client access to local libraries:

```
SMFLIBS =/usr/appl/common.lib|/usr/appl/client.lib
```

```
SMFLIBS =D:\dev\client.lib;D:\usr\my.lib;C:\dev\common.lib
```

To open remote libraries, SMFLIBS should include the host name on which the libraries reside; for example:

```
SMFLIBS =host!client.lib;host!server.lib;host!common.lib
```

Refer to the functions sm_l_open and sm_r_window for information on opening libraries at runtime.

Note: Separators are platform-specific. The vertical bar (|) can generally be used on any platform. A semi-colon (;) can also be used on Windows; a colon (:) on UNIX. Panther defines path name and command line argument separators for each platform in smcommon.h.

SMIBMVJAVA

Specifies the path for IBM's Visual Age for Java program.

SMIBMWSADMIN

Specifies the path for IBM's Administrative Console program. For Windows, if SMIBMWSADMIN is not set, it will look in the registry settings for WebSphere.

`SMINITJPL`

Specifies the JPL files to open in the server's `client.lib` library. On startup of web applications, the web application broker runs this file's `web_startup` procedure. On shutdown, the application runs the file's `web_shut` down procedure.

Use the `SMINITJPL`-specified JPL for two reasons:

- `web_startup` can declare any JPL globals required by the Web application.
- In three-tier configurations, `web_startup` initializes the Web application server as a client by calling `client_init`; `web_shutdown` calls `client_exit` to disconnect from the middleware.

In two-tier configurations, `web_startup` establishes database connections; `web_shutdown` cancels all connections.

`SMJAVACOMPILE`

Specifies the command used to compile Java. If it is unspecified, it defaults to

```
javac -deprecation "%s"
```

On Windows, the default value is overridden by a setting in `smvars.bin`:

```
cmd /c javac "%s" || pause
```

`SMJAVAEDITOR`

Specifies the editor to use when writing Java programs. If not specified, it defaults to the text editor defined in `SMEDITOR`.

`SMJAVAFACTORY`

Specifies the name of the class factory to use for Java programs. If not specified, it defaults to `com.prolifics.jni.ClassTagFactory`.

`SMJAVALIBRARY`

(optional) Specifies the location of Java libraries. If set in the environment, the specified library will override the default location.

`SMJVMOPT`

Specifies the options for the JVM (Java Virtual Machine).

`SMKEY`

Identifies the binary file containing a key translation table for your terminal. You can include a terminal mnemonic in the entry that matches the terminal type designated in your `SMTERM` variable or system `TERM` variable.

```
SMKEY = (vt100) $SMBASE/config/vt100keys.bin
```

SMLDBLIBNAME

Names the libraries whose contents are to be used by sm_ldb_init for local data block (LDB) initialization. The library files are loaded and activated at application startup. To specify multiple libraries, separate their names with a vertical bar (|) or colon, a semicolon under Windows, or the convention used by your operating system. Refer to "Using Local Data Blocks," on page 25-7 in *Application Development Guide* for information on using LDBs.

```
SMLDBLIBNAME = cust.lib|const.lib
```

```
SMLDBLIBNAME = my.lib:/u/dev/cust.lib
```

or under Windows:

```
SMLDBLIBNAME = c:\graphics.lib;c:\my\dev.lib
```

SMLDBNAME

Names the screens to be used by sm_ldb_init for local data block (LDB) initialization. To specify multiple libraries, separate their names with a vertical bar (|) or colon (:) under UNIX, a semicolon under Windows, or the convention used by your operating system. The listed files are loaded and activated at application startup after any libraries specified with SMLDBLIBNAME are loaded. Refer to "Using Local Data Blocks," on page 25-7 in *Application Development Guide* for information on using LDBs.

```
SMLDBNAME = ldb1.scr|ldb2.scr|ldb3.scr
```

SMMSGS

Identifies the binary file that contains messages and other printable strings used by the Panther runtime system and utilities. This variable is set to the full path name of the binary message file. For example:

```
SMMSGS     = $SMBASE\config\msgfile.bin
```

Refer to "Using Message Files," on page 45-2 in *Application Development Guide* for details about message files, and to library functions sm_msg_get and sm_msg_read.

SMPATH

Lists the directories where Panther should search for Panther files at runtime, such as screens, JPL procedures, and JDB databases. Place a vertical bar (|) between directory paths; do not include blank spaces.
You can include a terminal mnemonic in the SMPATH entry that matches the terminal type designated in your SMTERM variable or system TERM variable. For example, this entry specifies the search path for applications running on UNIX:

```
SMPATH = $SMBASE/config
```

An application server's environment file (such as `proserv.env`) can also set `SMPATH` to any directories that the server needs to access besides the application directory. If the environment file contains an `SMPATH` entry, make sure that the path includes `${SMBASE}/config`.

For example, the following entry in a UNIX server environment file ensures that servers using this file have access to files in `/u/myapps` and `${SMBASE}/config` as well as in the application directory:

```
SMPATH=/u/myapps:${SMBASE}/config
```

Note: Because a server's application directory is already set through its machine's configuration, `SMPATH` must not include the same directory again.

You can change `SMPATH` at runtime with `sm_soption`.

`SMPROVIDERURL`

(WebSphere applications only) Specifies the URL of the server running Web Sphere Application Server. Set in a runtime client environment in order to access the EJBs, as in:

`iiop://machineName:portNumber`

`SMRBCONFIG`

(JetNet/Oracle Tuxedo only) Specifies the middleware configuration file to use when a three-tier application activates. This variable must be set in the environment in which you boot an application. Set `SMRBCONFIG` to the full path name of the middleware configuration file—for example, `/usr/myapp/broker.bin`.

Also set this variable for a native client (a client that resides on an master or non-master machine) so it can connect to the middleware. Make sure that the `SMRBCONFIG` setting matches the value specified for the machine's local configuration file (refer to "Local JetNet Configuration File," on page 3-14 in *JetNet/Oracle Tuxedo Guide*).

Note: Applications using the Oracle Tuxedo middleware adapter can use `SMRBCONFIG` instead of `TUXCONFIG`.

Panther uses `SMRBCONFIG` as the default value on the Connect dialog when you ask to connect to the request broker from within the screen editor.

`SMRBHOST`

(JetNet/Oracle Tuxedo only) Together with `SMRBPORT`, this variable tells a workstation client how to connect to a server machine. `SMRBHOST` provides the request broker with the network addresses of the machines to which the client can connect.

Set this variable in the client's initialization file. Specify the value as a comma-separated list of one or more host names (or IP addresses in dot notation, for example, `192.200.3.42`), in a left-to-right order of precedence. For Oracle Tuxedo, this value is used to construct the `WSNADDR` string.

If the number of host name entries does not match that in `SMRBPORT`, the last entry in the shorter list is used to pair with the entries in the longer list. For instance, in the following example there are three host names and only two port numbers:

```
SMRBHOST = aspen,fir,willow
SMRBPORT = 300,400
```

Therefore, `aspen` uses port 300, while both `fir` and `willow` use port 400.

The value you provide is displayed by default on the Connect dialog when you choose to connect to the request broker from within the screen editor. If more than one host name is listed, a connection is attempted for each host, in order of precedence, until a connection is made.

`SMRBPORT`

(JetNet/Oracle Tuxedo only) Together with `SMRBHOST`, this variable tells a workstation client how to connect to a server machine. `SMRBPORT` provides the request broker with the port numbers associated with the machines (`SMRBHOST`) to which the client can connect.

Set this variable in the client's initialization file. Specify the value (within the range of 32k and 64k) as a comma-separated list of one or more numbers, in a left-to-right order of precedence, corresponding to the port number associated with the host machines specified in `SMRBHOST` (obtain port numbers from the person responsible for configuring the application).

For the Oracle Tuxedo middleware adapter, this value is used to construct the `WSNADDR` string.

If the number of entries in `SMRBPORT` does not match that specified in `SMRBHOST`, the last entry in the shorter list is used to pair with the entries in the longer list. Refer to `SMRBHOST` for an example.

The value you provide is displayed by default on the Connect dialog when you choose to connect to the request broker from within the screen editor. If more than one port number is listed, a connection is attempted for each, in order of precedence, until a connection is made.

SMSETUP

Identifies an binary setup file, in addition to the SMVARS file, that contains application variable settings. If the SMVARS and SMSETUP files set the same variables, the settings in the SMSETUP file take precedence. Use the SMSETUP file for variable settings that are specific to a particular terminal-type, project, or user.

```
SMSETUP = (xterm) xsetup.bin
```

SMTERM

Defines the terminal type. Set this variable if you want Panther to recognize a terminal type mnemonic that is different from the one that is defined in the system TERM variable.

For example, while other programs might run on a terminal in VT100 emulation, you want Panther to use the features of VT220 emulation; so while TERM is set to VT100, you can set SMTERM = vt220.

Users of Motif in X Windows can set SMTERM = X. To use Panther under Windows, set SMTERM = mswin.

SMTPCLIENT

(WebSphere initialization only) Specifies whether Oracle Tuxedo connectivity will be enabled in the application. If unset, the shared libraries needed for Oracle Tuxedo will not be loaded. If Oracle Tuxedo libraries are needed, set this variable in the global section of panther.ini to native or workstation, for native or workstation clients respectively.

SMTPCLIPFILE

(JetNet/Oracle Tuxedo only) In character mode, identifies the clipboard file to be used by the JIF editor. If not set, file clip.dat will be used.

SMTPINIT

(WebSphere initialization only) Specifies the default arguments to the client_init command for Oracle Tuxedo operations in WebSphere applications. Set this variable as part of the global settings in panther.ini.

SMTPJIF

(JetNet/Oracle Tuxedo only) Identifies the JIF file to open on application startup. This file must be in an open library, by default common.lib. In a

development environment, this variable determines which file to open initially when the JIF editor is invoked.

SMTRACE

Supplies a command string to pass to the sm_trace function during startup. It is new in Panther 5.30. For example:

```
SMTRACE=NOJAVA FRAMES=100 TRACEFILE="c:\temp\forex.trc"
```

would disable reporting Java events; would allocate 100 frames for trace data in dumps; and would start using `c:\temp\forex.trc` as the trace file.

SMUSER

Enables concurrent access to multi-user libraries. In order to control access, Panther must know each user's name. Panther also uses the value in SMUSER as the default user name when it constructs service aliases (refer to "Using Service Aliases to Test Services" on page 5-8 in *JetNet/Oracle Tuxedo Guide*).

Panther looks for SMUSER in the environment or, under Windows, in the initialization file. On UNIX and Windows, Panther can identify a user from the environment if SMUSER is not set.

Panther looks for user identification in these sources, listed in descending order:

1. SMUSER
2. LOGNAME
3. USER (UNIX)
4. USERNAME (Windows)

If neither SMUSER nor USERNAME is set on Windows, it calls the function GetUserName to find a user name. If none of these sources or methods provides identification, Panther prompts the user to supply a user name.

SMVARS

Identifies the binary setup file, usually `smvars.bin`. Panther uses the SMVARS file to find other configuration files and setup information. Set this variable in the environment or in the Windows initialization file. If SMVARS is not explicitly set, Panther uses the value of SMBASE to find `smvars.bin` in the `config` directory or, if not there, in $SMBASE itself.

A typical SMVARS file on Windows might contain these settings:

```
SMKEY     =  $SMBASE\config\winkeys.bin
SMMSGS    =  $SMBASE\config\msgfile.bin
SMPATH    =  $SMBASE\config
SMEDITOR  =  notepad.exe %s.
SMCOLMAP=(mswin) $SMBASE\config\wincmap.bin
SMCOLMAP=(web) $SMBASE\config\webcmap.bin
```

To make an entry specific to a terminal type, include the terminal's mnemonic enclosed in parentheses—for example, (`mswin`) for Windows.

`SMVIDEO`

Identifies the binary file that contains character-mode video control sequences and parameters used by a character-mode Panther runtime system. You can include one or more terminal mnemonic in the entry that matches the terminal type designated in your `SMTERM` variable or system `TERM` variable.

```
SMVIDEO = (hp|hp2392) $SMBASE/config/hp2392Avid.bin
```

Refer to Chapter 7, “Video File,” for details about video files, and to the library function `sm_vinit`.

Note: Video files are not required for GUI platforms, and may not be included in the Panther distribution.

`SMVIEWER`

Specifies the viewer for output of reports created with ReportWriter report browser. If you create PostScript reports, set this to a PostScript viewer. If this variable is not set, then `SMEDITOR` is used.

Behavior Variables

This section describes variables that control the behavior of a Panther client application and how users interact with it. Panther internally sets these variables to default values, as indicated in their descriptions by (*d*).

The variables described here apply to applications that run on GUI and character-mode platforms. For information about behavior variables that apply only to character-mode, refer to Chapter 5, “Character Mode Settings.”

Note: Behavior variables that control the GUI interface are generally ignored by Web applications—for example, cursor appearance, toolbar display, and mouse cursor appearance. In cases like these, client behavior is controlled by the user's Web browser.

Cursor Appearance and Movement

These variables control how the cursor appears and moves. (*d*) indicates the default setting.

`IN_HARROW`

Sets horizontal arrow movement:

Note: Left and right arrow keys usually move the cursor to the left or right of the current field, or as indicated by the behavior variable. However, in horizontal scrolling arrays, left and right arrow keys move the cursor to the next occurrence and scroll the array to the next available occurrence. Therefore, these variable settings only control cursor movement in single-occurrence fields.

`OK_FREE`
: Cursor moves freely.

`OK_RESTRICT`
: The cursor moves left and right in the current field but does not leave the field.

`OK_COLM`
: The cursor is positioned to the closest field on the current line.

`OK_SWATH`
: Same as `OK_COLM`.

`OK_NXTLINE`
: The cursor is positioned to the nearest field in the column closest to the current column. Wrapping is observed, if set.

`OK_NXTFLD`
: The cursor is positioned to the field closest to the current line and column. The calculation uses the diagonal distance, assuming a 5 to 2 aspect ratio.

`OK_TAB` (*d*)
: Left-arrow backtabs to the end of the previous field, and right-arrow tabs to the first character in the next field. Wrapping is observed if set. The next and previous field properties are not observed.

`OK_TABNXT`
: Like `OK_TAB`, but the next field and previous field properties are observed.

`IN_VARROW`
: Sets vertical arrow movement:

  Note: Using the up and down arrow keys usually causes the cursor to move up or down to the next field, or as indicated by the behavior variable. However, using up and down arrow keys in scrolling arrays moves the cursor to the next occurrence, and causes the array to scroll to the next available occurrence. Therefore, these variable settings only control cursor movement in fields having a single occurrence.

  `OK_FREE`
  : Cursor moves freely.

  `OK_RESTRICT`
  : Vertical arrow keys ignored in current field.

  `OK_COLM`
  : The cursor is positioned to the nearest field that overlaps the current column. Wrapping is observed, if set.

  `OK_SWATH`
  : The cursor is positioned to the closest field that overlaps the swath containing the current field. Wrapping is observed if set.

  `OK_NXTLINE` (*d*)
  : The cursor is positioned to the nearest field whose line is closest to the current line. Wrapping is observed, if set.

  `OK_NXTFLD`
  : The cursor is positioned to the field nearest the current line and column.

`OK_TAB`
: Down arrow tabs to the first character in next field; up arrow backtabs to last character in the previous field. The next and previous field properties are not observed.

`OK_TABNXT`
: Like `OK_TAB`, but the next field and previous field properties are observed.

`IN_ENDCHAR`

Specifies treatment of last character in a widget whose `autotab` property is set to `PV_NO`:

`OK_ENDWRITE`
: Last character is repeatedly overwritten if the cursor is in overwrite mode and the widget's `autotab` property is set to `PV_NO`.

`OK_ENDBEEP` (*d*)
: Terminal beeps when user attempts to overwrite last character in a widget whose `autotab` property is set to `PV_NO`.

`IN_RESET`

Sets options for field-reset:

Note: `IN_RESET` is ignored on word-wrapped fields.

`OK_NORESET`
: Arrow keys can enter the middle of a field.

`OK_RESET`
: When field is entered, cursor always goes to first character position, based on justification and punctuation properties.

`OK_TO_END` (*d*)
: When field is entered, cursor always go to the first available blank after (or before in the case of right-justification) the data.

`IN_VALID`

Sets conditions for validation on field exit:

`OK_VALID`
: Validation is performed whenever field is exited (`NL`, `TAB`, `BACKTAB`, arrows, mouse click).

`OK_NOVALID` (*d*)
: Validation is performed only when TAB or NL is pressed. Using arrow keys or a mouse click to leave a field does not validate the field.

`IN_WRAP`
: Sets options for arrow wrapping:

`OK_WRAP` (*d*)
: Arrow keys wrap. Vertical arrows wrap from top to bottom. Right arrows wrap to the beginning of next line (or first line). Left arrows wrap to end of previous line (or last line).

`OK_NOWRAP`
: Arrow keys do not wrap. Terminal beeps if user tries to move the cursor past the edge of the active screen.

Numeric Format and Display

`DECIMAL_PLACES`
: Sets default decimal places for real number display:

`PLACES_VARIABLE` (*d*)
: Sets the default number of decimal places used by `sm_dtofield` to equal the number of significant digits in the number, to a maximum of 20. Panther uses `sm_dtofield` to display real numbers.

`number`
: Sets the default number of decimal places used by `sm_dtofield` to *`number`*.

`OCTAL_SUPPORT`
: Specifies to interpret numbers with leading zero as octals:

`OCTAL_SUPPORT_ON`
: Numbers with a leading zero are treated as octal unless they begin with `0x`, `0X`, `0b`, or `0B`.

`OCTAL_SUPPORT_OFF` (*d*)
: Numbers with a leading zero are treated as decimal unless they begin with `0x`, `0X`, 0b,or `0B`.

Text Selection Appearance

The following variables set attributes for text selected in a single or multiline text widget. Refer to Table 2-1 on page 2-4 for a list of attribute keywords.

TXT_SELECT_ATTR

Sets desired attributes for the selected text. These attributes are added to those already assigned to the text widget. This defaults to HILIGHT REVERSE.

TXT_SELECT_MASK

Masks any attributes that should not be added to the selected text. If you assign a color as the selected text attribute, add NORMAL_ATTR to TXT_SELECT_MASK. Attributes of the occurrence are used unless they are masked out.

Function Keys

SMINICTRL

Associates control strings with function keys.

Each Panther screen contains a table of control strings associated with functions keys. You can also set default control strings for specified function keys either in the SMVARS file or in the environment. Panther uses absence of a control string for a given function key.

By including multiple SMINICTRL entries in your SMVARS file (or in the environment), you can define system-wide actions for specific function keys. The syntax for including SMINICTRL variables is as follows:

```
SMINICTRL = function_key = control_string
```

For example:

```
SMINICTRL = PF1 = &system_help
SMINICTRL = PF2 = ^toggle_mode
SMINICTRL = PF7 = ^jm_keys SPGD
SMINICTRL = PF8 = ^jm_keys SPGU
```

You can change application control strings at runtime through the application property control_string. For more information, refer to “Multi-item Properties” on page 19-41 in *Application Development Guide*.

To disable a Panther-supplied default function key, map it to a control string function that does nothing.

Menus and Toolbars

These variables control the behavior and display of menu bars and toolbar items and their corresponding tooltips. You can define them in a setup file and at runtime with `sm_option`. (*d*) indicates the default setting.

`TOOLBAR_DISPLAY`
Enables or disables tool bar display:

`TOOLBAR_OFF`
Disables display of tool bars. Panther continues to update the underlying data structures.

`TOOLBAR_ON` (*d*)
Allows display of tool bars.

`TOOLTIP_DISPLAY`
Enables or disables tool tip text display:

`TOOLTIP_OFF`
Disables display of tool tip text.

`TOOLTIP_ON` (*d*)
Enables display of tool tip text.

Message Display

These variables control message display. You can define them in a setup file, in the environment, and at runtime with `sm_option`.

Note: The `BLANK` attribute keyword is ignored for messages.

`MESSAGE_WINDOW`
Controls when messages appear in a window:

`WHEN_REQUIRED` (*d*)
Messages appear on the status line unless they are too large to fit. Oversize messages appear in a window. (Messages never appear on the status line in GUI environments).

ALWAYS

Messages always appear in a window unless they are explicitly sent to the status line—for example, with sm_d_msg_line and sm_msg.

SMSGPOS

Sets position of message line:

number

Sets the position for the message line by specifying a single number (1 is the top line of the display). This variable is ignored if the terminal has a hardware status line.

SMSGBKATT

Sets background attributes for message line. The default is SMSGBKATT=B_BLACK. The keywords for SMSGBKATT take the format B_*color*. Refer to Table 2-1 on page 2-4 for a list of attribute keywords.

STEXTATT

Sets the default display attribute for field status text. Refer to Table 2-1 on page 2-4 for a list of attribute keywords. The default is STEXTATT=WHITE.

Note: If you change the attributes but do not specify a color, the default color becomes BLACK. For example, if you use the entry STEXTATT = BLINK, status messages display with the foreground attributes BLINK and BLACK. Then, if you use the default message line background (see SMSGBKATT), status messages are not visible because they have black text on a black background. Always specify a foreground or background color when setting attribute for text. If this is not convenient, you can set the variable SMSGBKATT to a color other than B_BLACK.

EMSGATT

Sets the display attributes of message text displayed with sm_femsg and sm_ferr_reset, and the tag portion of sm_fquiet_err and sm_fqui_msg messages. The content of the tag is specified in the message file entry SM_ERROR (the default is ERROR:). The default is:

```
EMSGATT=WHITE BLINK HILIGHT B_HILIGHT
```

Refer to Table 2-1 on page 2-4 or a list of attribute keywords.

If you change this variable without specifying a foreground color, the default foreground color becomes BLACK. Refer to the note on STEXTATT for more information.

QUIETATT

Sets the display attributes of message text displayed with sm_fquiet_err messages. The default is QUIETATT=WHITE. See EMSGATT for changing the attributes of the tag portion of these messages. Refer to Table 2-1 on page 2-4 for a list of attribute keywords.

If you change this variable without setting a foreground color, the default foreground color becomes BLACK. Refer to the note on STEXTATT for more information.

Message Acknowledgment

These variables control how your application responds to user input when a message appears. (*d*) indicates the default setting.

ER_ACK_KEY

Defines the error acknowledgment key. This variable's value must be specified explicitly, either as a number (in decimal, hex, or octal) representing an ASCII character, as an ASCII mnemonic (SP, SOH, ETX), as a quoted character ('.', '_'), or as a logical key defined in smkeys.h. The default is ER_ACK_KEY=' ', the space key. If you define a value other than the spacebar, refer to ER_SP_WIND.

ER_KEYUSE

Specifies whether to use or discard error acknowledgment key:

ER_NO_USE (*d*)

Forces acknowledgment of all window-displayed error messages with either ER_ACK_KEY, space bar, or OK; error messages that are displayed to the status line must be acknowledged with either ER_ACK_KEY or spacebar. The valid keypress is immediately discarded.

All invalid responses to a window-displayed error message cause the terminal to beep (through calls to sm_bel). All invalid key presses to a status line message cause Panther to display an error window or beep, depending on how ER_SP_WIND is set.

ER_USE

Any keypress acknowledges a status line error message; this setting has no effect on window-displayed error messages. The type-ahead buffer is flushed when the message is displayed, and the

acknowledging keypress is saved for data-entry. If you set this as the default, you can force users to acknowledge a message by putting %Md at the beginning of the message text. Refer to "Using Message Files" on page 45-2 in *Application Development Guide* for more information.

ER_SP_WIND

Reminds user to acknowledge message (valid only for status-line error messages):

ER_YES_SPWIND (*d*)

If ER_KEYUSE = ER_NO_USE and the user presses another key when ER_ACK_KEY is expected, a window appears. The default message is Please hit the space bar after reading this message from the message file entries SM_P1 and SM_P2. If you are using this option and a key other than the space bar for message acknowledgement, modify the message file entry SM_SP1.

ER_NO_SPWIND

If ER_KEYUSE = ER_NO_USE, and the user presses another key when ER_ACK_KEY is expected, the terminal beeps (by calling sm_bel). A visual bell can be used if the video file has a BELL entry.

Shifting and Scrolling

These variables can help you establish standards for handling scrolling and shifting arrays and fields. You can define them in a setup file, in the environment, and at runtime with sm_option. If you change these defaults at runtime, call sm_option before opening the screen. (*d*) indicates the default setting.

ZM_DISPLAY

Specifies zoom window size preference:

ZM_ONSCREEN

Displays expanded zoom window at the display size of the operating system window (for example, in the xterm or jterm window). Useful for character-mode applications.

ZM_MAXIMUM

Displays expanded zoom window at its maximum physical display. Useful for applications running in GUI environment.

ZM_SC_OPTIONS
Sets zoom scroll options:

ZM_NOSCROLL
No scroll expansion on arrays.

ZM_SCROLL
Scrolls the current array and display as many occurrences as possible.

ZM_PARALLEL (*d*)
Scrolls all parallel or synchronized arrays. Display as many occurrences as possible.

ZM_1STEP
Scrolls and shift in one step.

ZM_SH_OPTIONS
Sets zoom shift options:

ZM_NOSHIFT
No shift expansion. Fields shift, but no horizontal zooming takes place.

ZM_SCREEN (*d*)
Shifting arrays have as many on-screen elements as the previous form, which is the original form if ZM_SC_OPTIONS = ZM_NOSCROLL is used. Otherwise, ZM_SCREEN displays as many items as possible. All synchronized arrays are shifted together.

IND_OPTIONS
Sets shift/scroll indicator options:

IND_NONE
No indicators.

IND_SHIFT
Shift indicators only.

IND_SCROLL
Scroll indicators only.

IND_BOTH (*d*)
Shift and scroll indicators.

SCR_KEY_OPT
Sets scroll field priority:

SCR_NEAREST (*d*)
: Nearest scrolling array to current field scrolls when scrolling keys are used (PGUP and PGDN).

SCR_CURRENT
: Scrolling keys (PGUP and PGDN) have no affect unless current field is a scrolling array.

SB_OPTIONS
: Sets scroll options for virtual windows:

SB_NONE
: No scroll bars or corner arrows.

SB_BARS (*d*)
: Shows scroll bars.

SB_CORNERS
: Shows corner arrows.

IND_PLACEMENT
: Sets position of shift and scroll indicators:

IND_FULL (*d*)
: Full width of field.

IND_FLDENTRY
: Left or right corner, according to the field's justification.

IND_FLDLEFT
: Left corner of field.

IND_FLDCENTER
: Center of field.

IND_FLDRIGHT
: Right corner of field.

Filename Conventions and Search

The following variables control how Panther builds and parses file names.

FCASE
: Sets case sensitivity for filename searches. This variable's setting is recognized only on UNIX applications:

`CASE_INSENS` (*d*)
Ignore case when searching for a file.

`CASE_SENS` (*d*)
Use case when searching for a file.

`F_EXTREC`
Specifies whether to recognize screen and utility I/O file extensions. The default value is the value of `EXTMULTS` for the system as defined in `smcommon.h`.

`FE_IGNORE`
Ignore extensions.

`FE_RECOGNIZE`
Recognize extensions.

`F_EXTOPT`
Sets placement of screen and utility I/O file extensions:

`FE_FRONT`
The extension is before the filename.

`FE_BACK` (*d*)
The extension is after the filename.

`F_EXTSEP`
Specifies the character used as a screen and utility I/O file extension separator. The default is a period (.). A separator character can be a number (decimal, hex, or octal) which represents an ASCII character, an ASCII mnemonic (`SOH`, `ETX`), or as quoted character ('.', '_').

Display Attributes for Grouped Items

These variables control the attributes of the cursor and selected items in groups. You can define them in a setup file, in the environment, and at runtime with `sm_option`. Refer to Table 2-1 on page 2-4 for a list of attribute keywords.

`GA_CURATT`
Sets desired attributes for the occurrence under the cursor. These attributes are added to those already assigned. The default is `BLINK B_HILIGHT`.

GA_CURMASK
Masks any attributes that should not be added to the cursor attributes. If you are assigning a color as the cursor attribute, add `NORMAL_ATTR` to `GA_CURMASK`. Attributes of the occurrence are used if they are not masked out.

GA_SELATT
Sets desired attributes for a selected group occurrence. These attributes are added to those already assigned to the occurrence. The default is `HILIGHT REVERSE`.

GA_SELMASK
Masks the attributes that should not be added to the attributes for the selected group occurrence. If you assign a color to `GA_SELATT`, add `NORMAL_ATTR` to `GA_SELMASK`. Attributes of the occurrence are used if they are not masked out.

Panther Mouse Cursor Appearance and Behavior

These variables control mouse cursor appearance and behavior only when the mouse is being run by Panther. In most cases, the mouse is under control of the native environment and mouse behavior is determined by the environment, not Panther.

CLICK_TIME
Sets the maximum amount of time between two mouse clicks that defines a double mouse click. The default setting is `250 ms`.

MOUS_CRSR_CHAR
Sets the character used to display mouse cursor. The default setting is a block character.

MOUS_CRSR_ATTR
Sets the desired attributes for the occurrence under the cursor. These attributes are added to those already assigned to the occurrence. The default setting is `REVERSE`. Refer to Table 2-1 on page 2-4 for a list of attribute keywords.

MOUS_CRSR_MASK
Masks any attributes that should not be added to the cursor attributes. If you are assigning a color as the cursor attribute, add `NORMAL_ATTR` to `MOUS_CRSR_MASK`. Attributes of the occurrence are used if they are not masked out. Refer to Table 2-1 on page 2-4 for a list of attribute keywords.

Miscellaneous Variables

These variables control a variety of customization issues. They can be defined in a setup file or in the environment as well as at runtime with `sm_option`. (*d*) indicates the default setting.

`CHAR_VAL_OPT`
Sets keystroke filter validation option:

`CHAR_BEEP` (*d*)
Causes a beep if user keyboard input does not pass character validation as defined in the `keystroke_filter` property for a data entry field.

`CHAR_MSG`
Displays message if user keyboard input does not pass character validation as defined in the `keystroke_filter` property for a data entry field.

`CLOSELAST_OPT`
Specifies whether to exit base form without exiting application:

`OK_CLOSELAST` (*d)*
Allows base form to close without exiting the application.

`NO_CLOSELAST`
Application ends when user exits base form.

`DA_CENTBREAK`
Sets the default century for two-digit dates through a two-digit number between `00` and `99`. Use this option to specify the breaking year between the twentieth and twenty-first centuries when Panther formats two-digit year to four-digit year specifications.

This option lets you specify that all two-digit year entries less than the number specified should be in the twenty-first century. For example, if you specify 45. then all two-digit year entries between 00 and 44 indicate the years 2000 to 2044, while those between 45 and 99 indicate 1945 to 1999.

The default value is 50.

`ENTEXT_OPTION`
Sets how to process screen entry:

`LDB_FIRST` (*d*)
: LDBs are examined first for the value of a field, then the screen, on screen entry. On screen exit, this order is reversed.

`FORM_FIRST`
: Screen is examined first for the value of a field, then LDBs, on screen entry. On screen exit, this order is reversed.

`EXPHIDE_OPTION`
: Sets screen expose/hide options:

`OFF_EXPHIDE`
: Processes screen entry or exit functions only when screen is explicitly opened or closed.

`ON_EXPHIDE` (*d*)
: Processes screen functions when screen is explicitly opened or closed, when screen is exposed by closing an overlying window, or when screen is hidden by opening an overlying window.

`JAVA_USE`
: Sets the Java initialization option during Panther start-up:

`JAVA_IS_USED` (*d)*
: Initializes Java.

`JAVA_NOT_USED`
: Do not initialize Java. Set in development executables. For distributed executables, remove the call to `sm_java_init` in `jmain.c`, or edit the makefile to remove `SM_JAVA`.

`JAVA_USE_CODESET`
: Sets the Java String to codeset conversion option for Panther's Java APIs

`:CODESET_NOT_USED` (*d)*
: When passing String data to Panther's Java API, only the low order byte of each Unicode Character is used to form a single-byte-per-character character array, without regard for the value of the `codeset` application property. String data is returned by constructing a String such that each byte of application data forms the low order byte of a Unicode Character in the Java String, without regard for the current `codeset`. The high order byte is always set to 0.

CODESET_IS_USED
: Enables the codeset property to be used by Panther's Java API. Methods that take String arguments as parameters will encode them into the specified codeset for processing. Methods that return String instances will convert application data to be returned into Unicode in accordance with the specified codeset.

LISTBOX_SELECTION
: Enables extended selection in list boxes:

  EXTENDED_SELECTION (*d*)
  : Enables extended selection.

  SIMPLE_SELECTION
  : Allows only one selection at a time.

STARTSCREEN
: Specifies the name of the Panther screen that the application first displays. Panther searches open libraries. If no screen is found, Panther reports an error.

  STARTSCREEN is not supported in the application's initialization/resource file or in the environment. It must be set in the file specified by SMVARS or SMSETUP. You can also specify the start screen by setting start_screen_name in jmain.c or jxmain.c.

WWTAB
: Sets the number of spaces to move when the TAB key is pressed in a word wrap multiline text field. The default is 5. On Motif, this option is ignored.

WW_COMPATIBLE
: Determines whether word wrap arrays are to be validated when the user moves from one occurrence to another.

  WW_COMPATIBLE_OFF
  : Do not validate. This is the default.

  WW_COMPATIBLE_ON
  : Validate when the user moves from one occurrence to another within a word wrap array.

XMIT_LAST
: Sets NL or TAB to act like XMIT on last field of screen (triggering screen validation):

`XMIT_NL`
: NL (New Line) logical key behaves like `XMIT` on last field of a screen.

`XMIT_TAB`
: `TAB` logical key behaves like `XMIT` on last field of a screen.

`XMIT_NL_TAB`
: `NL` and `TAB` behave like `XMIT` on last field of a screen.

`XMIT_DISABLE` (*d*)
: `NL` and `TAB` respond as defined and do not emulate the `XMIT` key when used in the last field of a screen.

Sample Setup File

The following sample file illustrates the syntax for setting most of the variables discussed in this chapter. These variables can be designated in the environment or, as recommended, in a setup file, and can be altered at runtime with specific library functions.

```
SMKEY = (vt100 | x100) /usr/panther/config/vt100keys.bin
SMMSGS = /usr/config/msgfile.bin
SMPATH = /usr/app/forms|/usr/me/testforms
SMSETUP = hpsetup.bin
SMVIDEO = (vt100 | x100)/usr/panther/config/vt100vid.bin
SMINICTRL= PF2 = ^toggle_mode
SMINICTRL = PF3 = &popwin(3,28)
SMINICTRL = XMIT = ^commit all

IN_BLOCK = OK_NOBLOCK
IN_WRAP = OK_WRAP
IN_RESET = OK_NORESET
IN_ENDCHAR = OK_ENDWRITE
IN_VALID = OK_VALID
IN_VARROW = OK_FREE
IN_HARROW = OK_TAB
ER_ACK_KEY = PF12
ER_KEYUSE = ER_NO_USE
ER_SP_WIND = ER_YES_SPWIND
```

```
EMSGATT = RED; RED, REVERSE
QUIETATT = CYAN HILIGHT BLINK
STEXTATT = WHITE REVERSE
QMSGATT = CYAN REVERSE
SMSGBKATT = RED
SMSGPOS = 25
ZM_SC_OPTIONS = ZM_SCROLL
IND_OPTIONS = IND_BOTH
IND_PLACEMENT = IND_FLDRIGHT
SB_OPTIONS = SB_CORNERS
F_EXTREC = FE_RECOGNIZE
F_EXTOPT = FE_BACK
F_EXTSEP = '.'
DW_OPTIONS = DW_OFF
ENTEXT_OPTION = LDB_FIRST
FCASE = CASE_SENS
```

CHAPTER

3 Windows Initialization File

An application that runs under Microsoft Windows uses an initialization file to set defaults. Initialization file settings control the appearance and behavior of Panther applications run under Windows—for example, specifications for basic color mappings, window display and other behavior.

The initialization file that you supply with a deployed application can provide initial settings; individual users can later change these settings to suit their preferences. Initialization file settings override any duplicate settings in `SMVARS` or `SMSETUP` files.

Other Windows attributes such as the color scheme must be set from the Windows control panel. Control panel changes are written to the Windows initialization file. For more information, refer to the Windows documentation.

Note: Several initialization files are associated with Panther. However, this chapter only describes the one that is used to initialize Panther applications—`prol5w32.ini` or `prol5w64.ini`.

Initialization File Search

Each application can have its own initialization file. At startup, Panther looks for the initialization file in the following places, listed in descending order of precedence. You can use any of the following methods:

1. The initialization filename that is supplied as an argument to `sm_pi_mw_setup` in the file `piinit.c` for a Panther application executable, and to `sm_pi_mw_jxsetup` in the file `pijxinit.c` for a Panther development executable. To set an application's initialization file in this way, edit the appropriate source file, which is in the `link` directory, then recompile.

   In the default distribution, these functions are supplied an empty string argument.

2. The initialization file that is supplied as the `-ini` option used when starting the application. The `.ini` file extension is optional.

3. An initialization file whose base name is the same as the Panther executable—for example, `LEADS.INI` for the executable `LEADS.EXE`. The initialization file must be in the Windows directory if you do not supply a path.

4. The initialization file that is supplied with Panther (`prol5w32.ini` or `prol5w64.ini`) and installed in the Windows directory.

Initialization File Format

Initialization files are arranged as a list of variables; each variable takes a value in this format:

`variable=value`

For example, this entry sets the StackedWindowType variable to Dialog.

```
StackedWindowType=Dialog
```

Entries are organized into several sections; each section starts with a bracketed title:

- [Prolifics Options]—Settings for general Panther variables such as SMBASE, and variables for Windows-specific behavior and appearance.
- [Prolifics PaletteColors] and [Prolifics NoPaletteColors]—Maps Panther basic colors to GUI color values. Modern display adaptors support so many colors so that the use of color pallets is rarely needed.
- [Prolifics Fonts]—Configures the application's default font.
- [Prolifics StatusLine]—Configures how the status line appears.
- [Prolifics Help]—Specifies behavior of Panther interface to Windows Help.
- [Prolifics DDE]—Supports end-user DDE client links.
- [Prolifics EditMenu] and [Prolifics WindowsMenu]—Set the labels, mnemonics, and accelerators for the application's Edit and Windows menus in the Editor.

Sections and entries within each section can be in any order. Comments are indicated with a semicolon at the start of the line.

General Variable Settings

The initialization file can set a number of general application variables, which are described in Chapter 2, "Application Variables." These settings supersede duplicate settings in the Windows environment, thus allowing unique settings for different applications running in the same Windows environment. You can set these variables:

SMBASE
SMDICNAME
SMFLIBS

SMPATH
SMRBHOST
SMRBPORT
SMTERM
SMTPJIF
SMUSER
SMVARS

In this section, you can also set or override the LM_LICENSE_FILE environment variable setting. This variable can be set to a file or to a license server port—for example, LM_LICENSE_FILE=1700@myhost.

To use the environment settings for any of these variables, set it to an empty value. For example:

```
SMPATH=
```

Behavior and Display Settings

The following variables determine the GUI's appearance and behavior. They are set in the [Prolifics Options] section. *(d)* indicates the default setting.

Note: This section also typically sets a number of general setup variables such as SMBASE and SMVARS.

3D
: Set to Yes (the default) or No in order to enable or disable three-dimensional display of various application objects—message boxes, Windows common dialog boxes, and application screens. This setting can be overridden for individual screens through its 3d property. However, display of message boxes and Windows common dialog boxes is always set in the initialization file

FrameTitle
: Sets the title text in the MDI frame around a Panther application.

IconPosition
: Sets the location of the application's minimized icon on the screen's X and Y axes with this format:

`IconPosition=`*`x-position y-position`*

`x-position` and *`y-position`* are measured in pixel units.

`IntroPixmap`

Specifies the image that appears during application startup. Set this variable to the image's file name. Refer to "Displaying a Picture on Widgets" on page 21-10 in *Using the Editors* for supported image file types.

`KeepInMDI`

Determines initial placement of a window in relation to the MDI frame:

`Yes`

Forces initial placement of a newly created window within the MDI frame, if its size allows.

`No` *(d)*

Allows initial placement of the window anywhere within the MDI frame.

`MDIWallpaperPixmap`

Specifies a wallpaper image. Set this variable to the image's file name. Refer to "Displaying a Picture on Widgets" on page 21-10 in *Using the Editors* for supported image file types.

`MDIWallpaperStyle`

Sets the position of a wallpaper image on Panther's MDI parent window:

`Center`

Centers the image on your Panther MDI screen.

`Tile` *(d)*

Places the image in the upper left corner of the window; the image is repeated as many times as necessary to fill the entire window.

`MoveThreshold`

Specifies in pixels the distance that the cursor must be moved in order to be considered a drag instead of a click. This applies when dragging the rubberband, or creating, moving, or resizing an object.

`NormalPosition`

Sets the location of the application's MDI frame on the screen's X and Y axes when it is restored, with this format:

`NormalPosition=`*`x-position y-position`*

`x-position` and *`y-position`* are measured in pixel units.

`NormalSize`
: Sets the size of the MDI frame in when it is restored, with this format:

  `NormalSize=`*`width height`*

  width and *height* are measured in pixel units.

`PaletteUse`
: Specifies whether to use the color palette to produce colors:

  `Enable` *(d)*
  : Panther uses the color palette to produce colors, allowing the full range of RGB colors supported by your display device without having to make use of dithering. If `PaletteUse` is set to `Enable`, Panther uses the color mappings in the [`Prolifics PaletteColors`] section.

  `Disable`
  : Panther does not use the color palette. An application is limited to the reserved system colors (20 on a 256-color display) and dithered combinations of them. This setting requires fewer resources. Set `PaletteUse` to `Disable` only if you need to run other applications that use colors intensively.

    If `PaletteUse` is set to `Disable`, Panther uses the color mappings in the [`Prolifics NoPaletteColors`] section.

`SaveStateOnExit`
: If set to `Yes` (the default), the size and location of the MDI frame is saved between sessions.

`StackedWindowType`
: Set to either `Dialog` or `MDI`:

  `Dialog` *(d)*
  : Forces all stacked windows in a Panther application to behave as dialog boxes: they are modal, have a dialog-style border, and can move outside the MDI frame. This can cause unpredictable behavior if the application moves stacked and sibling windows on the window stack, or if the application creates or breaks sibling relationships between windows after they are displayed.

  `MDI`
  : Stacked windows behave as regular MDI windows.

`StartupState`
: Set to `Minimized`, `Normal` (the default), or `Maximized`. This setting controls the size of the MDI frame when Panther starts. Windows determines the initial size if no value is given.

Basic Color Mappings

Panther defines sixteen basic colors—eight highlighted and eight unhighlighted. These colors are displayed on the screen editor's color palette (refer to "How to Set an Object's Color with the Color Palette" on page 11-4 in *Using the Editors*):

`black`	`red`	`hi_black`	`hi_red`
`blue`	`magenta`	`hi_blue`	`hi_magenta`
`green`	`yellow`	`hi_green`	`hi_yellow`
`cyan`	`white`	`hi_cyan`	`hi_white`

The initialization file can map these colors to any of the colors supported by Windows. Panther basic colors are mapped in one of two sections, depending on whether the Windows color palette is enabled or disabled (refer to page 3-6, "`PaletteUse`").

If the Windows color palette is enabled (`PaletteUse=Enable`), basic colors are mapped to Window colors in the [`Prolifics PaletteColors`] section.

- If the application runs on a display device that does not support a color palette or the color palette is disabled (`PaletteUse=Disable`), basic colors are mapped in the [`Prolifics NoPaletteColors`] section.

Entries in both sections use this syntax:

`pantherColor=rgbValue`

`pantherColor` is one of the basic Panther colors listed earlier. *`rgbValue`* is an RGB value in the form *`red/green/blue`* where *`red, green, and blue`* are numbers between 0 and 255. For example:

```
blue=0/0/255
```

Under Windows, foreground colors must be primary colors—dithered patterns are invalid. If you specify a non-primary color, Windows rounds it up to a primary color.

You can use the Windows palette feature (on the Windows Control Panel) to interactively mix your colors, then note the values and transfer them to the initialization file.

Note: To create custom color aliases beyond the sixteen basic colors, use the configuration map file—refer to "Configuration Map File" on page 45-25 in *Application Development Guide*.

Status Line Appearance

The [`Prolifics StatusLine`] section contains several variables that determine the appearance of the status line appears. By default, the status line is an etched out bar with the same colors used for push buttons. *(d)* indicates the default setting.

`Flags`

Set this variable to one or more values:

`ETCHEDIN` *(d)*

Causes the status bar to appear etched into the MDI frame rather than etched out.

`PURECOLORS`

Allows only non-dithered colors to paint the status bar.

`STATUSBARCOLORS`

Enables the four status bar color variables (`BackColor`, `HighLightColor`, `ShadowColor`, and `TextColor`). If this flag is not set, the control panel color settings for buttons are used.

`ShadowWidth`

Sets the width of a three-dimensional status bar's shadow, in pixel units. The default width is 2 pixels. To avoid a three-dimensional effect, set this variable to 0.

`BackColor`
`HighLightColor`
`ShadowColor`
`TextColor`

These variables set colors for the status bar: default background color, text color, shaded highlights, and darkened shadow for the status line. Settings for these color variables are used only if the `Flags` variable is set to `STATUSBARCOLORS`. Set these variables to an RGB value with this format:

`variable=red/green/blue`

`red`, *`green`* and *`blue`* are numbers between 0 and 255. For example:

```
blue=0/0/255
```

You can use the Windows palette feature on the Windows control panel to mix colors interactively, then note the values and enter them as shown above.

For compatibility with other Windows applications, background color defaults to grey. Other color default values are calculated by `sm_ManipulateColor`. Messages with embedded display attributes can override the default background color.

Help Behavior

The [`Prolifics Help`] section specifies the behavior of the Panther interface to the Windows help engine.

`EditorHelpPath`

Sets the path to the directory that contains screen editor online help. The default installation places the help files in `$SMBASE\docs\books`.

`HelpFile`

Set this variable to invoke Windows help. This variable should be set to the path of the desired Winhelp database file. The path can be a fully specified or it can be a file name that is on the path specified by the `SMPATH` variable.

DDE

This section supports DDE client and server links. For general information on setting DDE client/server links between Panther and other applications, refer to Chapter 46, “Dynamic Data Exchange,” in *Application Development Guide*.

`DDEServer`
: Set to `On` or `Off`. When set to `On`, this variable enables Panther as a DDE server.

`DDEClient`
: Set to `On` or `Off`. When set to `On`, this variable enables Panther as a DDE client.

`screenname ! fieldname = service | topic ! item`
: Specifies hot links to server applications. The format for server, topic, and item arguments is specific to the server application. For example, a link to a Quattro Pro spreadsheet might look like this:

```
salesScr!totalSales=QPW|C:\myAcct\sales.wb1!$A:$A$10..$A$10
```

Table 46-1 on page 46-7 in *Application Development Guide* shows the syntax used by three widely used Windows applications. Refer to the server application's documentation for information on server argument formats. For more information on setting Panther as a DDE client, refer to “Panther as a DDE Client” on page 46-5 in *Application Development Guide* .

Edit and Windows Menus

Two sections, [Prolifics EditMenu] and [Prolifics WindowsMenus], let you set the labels, mnemonics, and accelerators for the Edit and Windows menus, respectively. For example, this entry sets the Cascade item label and specifies its first letter as the item's mnemonic (&C):

```
CascadeLabelAndMnemonic=&Cascade
```

Note: If you change the default settings for accelerators, add the corresponding items in the accelerator table, in the mwjxform.rc file.

Sample Initialization File

```
[Prolifics Options]

;
;   The following are various general Prolifics options.
;   Each option is preceded with a comment that indicates
;   what it is for.

;   Set this option to the name you want in the MDI Frame
;   Window's title bar. You should set this to an appropriate
;   value for your application.
FrameTitle=Panther

;   Name of the graphic file to display upon startup of the
;   application.
IntroPixmap=

;   SMBASE, SMVARS, SMPATH, SMUSER, SMRBHOST and SMRBPORT may
;   be set here to override the environment variable
;   settings. This permits different settings for different
;   applications running in the same environment, without
```

```
;   needing to restart Windows. Setting any of these options
;   to an empty value will maintain the environment settings.
SMBASE=C:\Program Files\Prolifics\Panther5
SMVARS=
SMPATH=
SMUSER=
SMRBHOST=
SMRBPORT=

;   Use this option to override the SMFLIBS environment
;   variable under Windows. If the libraries are accessed
;   remotely, change the syntax of the libraries to include
;   the host name (e.g.
;   SMFLIBS=host!client.lib;host!common.lib;host!server.lib).
SMFLIBS=client.lib;common.lib;server.lib

;   Use this option to override the SMDICNAME environment variable
;   under Windows.  If the libraries are accessed remotely,
;  uncomment this line and use the host name of the remote system.
;SMDICNAME=host!data.dic

;   Use this option to override the SMTERM environment variable
;   under Windows.
SMTERM=mswin

;   Use this option to set or override the LM_LICENSE_FILE
;   environment variable setting.  Note that this can be set to
;   a file or to a license server port (e.g. 1700@myhost).
LM_LICENSE_FILE=

;   Setting this option to "No" will cause the application to
;   lose the 3-D effect for all user screens and system
;   dialogs. Defaults to "Yes" if unspecified.
3D=Yes

;   The following options control the size and location of
;   the MDI frame when Prolifics starts. If no values are
;   given, the initial size & location will be determined by
;   Windows.

;   Set this option to "Minimized", "Normal" or "Maximized".
;   Defaults to "Normal".
StartupState=Normal

;   This option sets the normal (i.e. "Restored") size (width
;   height) of the MDI frame in pixels. For example,
;   NormalSize=100 150 makes the MDI frame 100 pixels wide
;   and 150 pixels high when in a restored state.
NormalSize=
```

```
;   This option sets the normal (i.e. "Restored") location
;   (x y) of the MDI frame in pixels.
NormalPosition=

;   This option sets the location (x y) of the application's
;   minimized icon in pixels. Use this to override Windows'
;   default placement.
;IconPosition=

;   Set this option to "Yes" to save the size and location of
;   the MDI frame in this file each time you exit Prolifics. Any
;   uncommented settings for StartupState, NormalSize,
;   NormalPosition and IconPosition are overridden upon
;   exiting. The default is "No" if unspecified.
SaveStateOnExit=Yes

;   Name of the graphic file to display as the MDI frame
;   background. Style determines if graphic uses a "Tile"
;   (default) or "Center" style.
MDIWallpaperPixmap=
MDIWallpaperStyle=

;   The next option controls how Prolifics makes use of the
;   color palette, if your display device supports one. Use
;   of the palette allows the full range of RGB colors
;   supported by your display device to be displayed without
;   dithering. If several applications which use the color
;   palette are running, they share the palette. The
;   application which is active is guaranteed to display its
;   colors correctly. If applications share the palette
;   correctly, applications which are not active get colors
;   which are as close as possible to the requested colors.
;   More information on color palettes can be found in
;   chapter 19 of the Windows SDK "Guide to Programming".

;   If set to Disable, Prolifics does not make use of the
;   color palette. Your application is limited to the
;   reserved system colors (20 on a 256 color display) and
;   dithered combinations of them. This setting requires
;   fewer resources. In this configuration, the colors
;   displayed by Prolifics are independent of the focus. You
;   should only set this option to Disable if you tend to run
;   other applications that are very color intensive.

;   Set this option to "Enable" or "Disable". The default is
;   "Enable".
PaletteUse=Enable

;   The following option can be used to force all stacked
;   windows in a Prolifics application to be dialog boxes.
```

```
;   This makes them modal, gives them a dialog style border,
;   and allows them to move outside of the MDI frame. Note
;   that this may lead to unpredictable behavior if the
;   application moves stacked and sibling windows around on
;   the window stack, or if the application "siblingizes"
;   and/or "desiblingizes" windows after they are displayed.
;
;   Set this option to "Dialog" to force stacked windows to
;   be dialog boxes or to "MDI" if you want to use regular
;   MDI windows.
StackedWindowType=Dialog

;   The option "MoveThreshold" is the number of pixels which
;   the mouse must be moved in order to be considered a drag
;   instead of a click. This applies when dragging out a
;   rectangle to create an object, or when trying to move or
;   resize an object. If you actually do want to move or
;   resize an object by less than this amount, drag a larger
;   distance first, then drag back to the desired position.
;   (Do not release the mouse in between.) This only takes
;   effect in Edit Mode.
MoveThreshold=5

;   Setting this option to "Yes" will cause new MDI windows
;   to be positioned within the MDI frame, if possible. "No"
;   (the default) places no constraints on the initial
;   position of windows.
KeepInMDI=No

;   The following two options can be used to disable
;   Prolifics as a DDE client or server. The default is for
;   Prolifics to be enabled as both a client and server. Set
;   these to "Off" to disable either or both.
DdeClient=On
DdeServer=On

[Prolifics PaletteColors]

;   This section is used to match the sixteen Prolifics Basic
;   colors to RGB values. It is not optional, as Prolifics
;   must have a way to perform this mapping. The 16 colors
;   listed here correspond to the colors of the Screen
;   Editor's Color Palette screen. You may change any of
;   these values to override the default. The values for
;   these colors are decimal RGB values. You may run Control
;   Panel - Colors to look at colors by decimal RGB value.

black=128/128/128
blue=0/0/191
green=0/191/0
```

```
cyan=0/191/191
red=191/0/0
magenta=191/0/191
yellow=191/191/0
white=191/191/191
hi_black=0/0/0
hi_blue=0/0/255
hi_green=0/255/0
hi_cyan=0/255/255
hi_red=255/0/0
hi_magenta=255/0/255
hi_yellow=255/255/0
hi_white=255/255/255

[Prolifics NoPaletteColors]

;   The colors in this section will be used if the display
;   device does not support a color palette, or you have
;   disabled the use of the color palette using the
;   PaletteUse option.

;   This section is used to match the sixteen Prolifics Basic
;   colors to RGB values. It is not optional, as Prolifics
;   must have a way to perform this mapping. The 16 colors
;   listed here correspond to the colors of the Screen
;   Editor's Color Palette screen. You may change any of
;   these values to override the default.The values for these
;   colors are decimal RGB values. You may run Control Panel
;   - Colors to look at colors by decimal RGB value.

black=128/128/128
blue=0/0/128
green=0/128/0
cyan=0/128/128
red=128/0/0
magenta=128/0/128
yellow=128/128/0
white=192/192/192
hi_black=0/0/0
hi_blue=0/0/255
hi_green=0/255/0
hi_cyan=0/255/255
hi_red=255/0/0
hi_magenta=255/0/255
hi_yellow=255/255/0
hi_white=255/255/255

[Prolifics Fonts]
```

```
;   The "Font" option sets the application-wide default font.
;   It can be a font description in the form:
;   fontname-size[-bold][-italic][-underline], or it can be a
;   GUI-independent font (see the [Prolifics FontTable] section
;   below).  You can also use one of the system fonts below:

;       SYSTEM_FONT       -- the font Windows uses in menus, etc.
;       SYSTEM_FIXED_FONT -- a fixed pitch font
;                            (used in draw mode with SYSTEM_FONT)
;       OEM_FIXED_FONT    -- the PC, MSDOS character set
;                           (line drawing will work with this one)
;       ANSI_FIXED_FONT   -- Courier fixed pitch
;       ANSI_VAR_FONT     -- Helvetica variable pitch
;       DEFAULT_GUI_FONT  -- font used form menus, dialog boxes,
;                             etc.  (Windows 95 only)

;   If not specified, the default is SYSTEM_FONT in Windows NT and
;   DEFAULT_GUI_FONT in Windows 95.
;Application Font=SYSTEM_FONT

[Prolifics StatusLine]

;   This section configures how the status line appears.
;   By default, the status line is an etched out bar with the
;   same colors that Windows uses for push buttons.
;   The following options modify its appearance:

;   The "Flags" option sets flags for the status line.
;   Available flags are:

;   STATUSBARCOLORS - If this flag is set, the options
;                BackColor, TextColor, HighLightColor, and
;                ShadowColor below are meaningful. If it is not
;                set, colors used for the status bar are those
;                 set for buttons in the control panel.

;   PURECOLORS - All colors used for painting the status
;                bar are forced to pure non-dithered colors.
;   ETCHEDIN   - The status bar is etched in to the MDI
;                frame, rather than being etched out. Etched out
;                is the default.
;
;   By default, no flags are set.
Flags=

;   The "ShadowWidth" option sets the width of the 3D status
;   bar shading in pixels. It defaults to 2. If set to 0,
;   there is no 3D effect.
ShadowWidth=2
```

```
;   The "BackColor", "HighLightColor", and "ShadowColor"
;   options are the colors used to paint the status bar
;   background, shaded highlights, and darkened shadow.
;   These options only take effect if STATUSBARCOLORS is
;   one of the flags specified in the "Flags" option above.
;   The colors can be GUI independent colors. "BackColor"
;   defaults to gray. "TextColor", "HighLightColor" and
;   "ShadowColor" are calculated using sm_ManipulateColor()
;   if they are not specified here.

BackColor=128/128/128
;TextColor=0/0/0
;HighLightColor=255/255/255
;ShadowColor=0/0/0

;   The "Font" option sets a font for use on the status line.
;   It can be one of the system fonts, like SYSTEM_FONT, or
;   ANSI_VAR_FONT, or it can be a font description in the
;   form: fontname-size[-bold][-italic][-underline] or it
;   can be a GUI-independent font. It defaults to
;   SYSTEM_FIXED_FONT.

Font=MS Sans Serif-10
;
[Prolifics Help]
;
;   This section specifies the behavior of the Prolifics
;   interface to external help engines.

;   This option contains the name of the help file to be used
;   by the external help system you've installed.
HelpFile=

;   The option "EditorHelpPath" specifies a path where the
;   directory that contains the DynaText book directory can be
;   found. This should be removed when providing an .ini file
;   with your finished application. (until Panther 4.2 only)
EditorHelpPath=$SMBASE\docs\books

[Prolifics DDE]

;   This section supports end user DDE client links. Specify
;   the Prolifics screen name, Prolifics widget name, the
;   remote server name, the topic, and the item.

;   Example:
;        scrnname.scr!widgetname=Excel|Sheet1!R1C1

;   The screen name MUST be separated from the widget name
;   with an '!'. The Server, Topic and Item MUST be separated
;   with '|' and '!' characters.
```

```
;WARNING: QuattroPro requires full path name for its topic.
;         For example, if we connect linkfunc.scr client
;         topic and DDETextField client item with QPW server,
;         FROMINI.WB1 server topic, and A1 server item, we
;         write:
;      linkfunc.scr!DDETextField=QPW|C:\QPW\FROMINI.WB1!A1

;NOTE: Prolifics topic and item can be specified in lower,
;      upper, or mixed cases. QuattroPro and Microsoft Excel
;      servers allow specification of names of their servers,
;      topics, and items in lower, upper, or mixed cases.

[Prolifics EditMenu]

;  This section enables a user to set Edit Menu labels,
;  mnemonics, and accelerators. Below are examples of
;  default settings which are commented out.
;  WARNING: in case a user changes default settings for
;           ACCELERATORS, he/she should change corresponding items
;           in accelerator table prorun.rc and prodev.rc file.

;CutLabelAndMnemonic=Cu&t
;CutAccelerator=CtrlDel+X
;CopyLabelAndMnemonic=&Copy
;CopyAccelerator=Ctrl+C
;PasteLabelAndMnemonic=&Paste
;PasteAccelerator=Ctrl+V
;DeleteLabelAndMnemonic=Delete
;DeleteAccelerator=Del
;ClearLabelAndMnemonic=Clear
;ClearAccelerator=
;SelectAllLabelAndMnemonic=&Select All
;SelectAllAccelerator=

[Prolifics WindowsMenu]

;  This section enables a user to set Windows Menu labels,
;  mnemonics, and accelerators. Below are examples of default
;  settings which are commented out.
;  WARNING: in case a user changes default settings for
;           ACCELERATORS, he/she should add corresponding items
;           in accelerator table in prorun.rc and prodev.rc files.

;CascadeLabelAndMnemonic=&Cascade
;CascadeAccelerator=
;TileLabelAndMnemonic=&Tile
;TileAccelerator=
;ArrangeIconsLabelAndMnemonic=Arrange &Icons
;ArrangeIconsAccelerator=
```

```
[Environment]

; The set of directories or JAR files where the users'
; class files are.
; CLASSPATH=

[databases]

installed=odbc

[dbms odbc]
description=Microsoft ODBC version 3
driver=odb3dm32.dll
model=tmodb132.dll
```

CHAPTER

4 Setting Motif Defaults

Applications that run under Motif use resource files to set defaults for the GUI. The resource file controls how Motif and the application running under Motif appear and behave. You can set up the initial state; users can later change these settings to suit their preferences.

Resource Files

Resource preferences are indicated by setting attribute/value pairs.

Panther applications use resource files to determine values for a variety of attributes. These include:

- Default colors
- Mapping Panther basic colors to Motif colors
- GUI-independent color names
- Application behavior

Resource Filenames

Each application can have an application-specific resource file. The name of this file is determined by the class name for the application. The class name for a Panther application is set by the argument to Panther's GUI initialization function. To change the class name for a Panther application executable, edit the argument supplied to the function `sm_pi_xm_setup` in `piinit.c`; for a Panther development executable, edit the argument to `sm_pi_xm_jxsetup` in `pijxinit.c` . These source files are in the distribution's `link` directory.

At initialization, the main routine of a Panther application (usually either `jmain.c` or `jxmain.c`) calls either the function `sm_pi_init` or `sm_pi_jxinit` to initialize the GUI. These routines in turn call `sm_pi_xm_setup` or `sm_pi_xm_jxsetup`, which sets the class name.

The default class name for all Panther software tools in the distributed software is `Prolifics`. Therefore the application-specific resource filename is `Prolifics`.

Resource File Format

Under Motif, resource file entries are formatted as colon separated attribute/value pairs:

`attribute:value`

The value is understood to be any text to the right of the colon. White space between the colon and `value` is ignored. In the following example, the attribute `stackedWindowsAreDialogs` is set to `false`:

```
Prolifics*stackedWindowsAreDialogs:     False
```

In this example, `Prolifics` is the class name. It restricts this resource to applications of the class `Prolifics`. Resources can be further restricted to screens and even to individual widgets. The class name for a Panther application is determined at application initialization. You can also specify an instance name for an application via the standard Xt command line switch `-name`.

Comments are indicated by starting the line with an exclamation point. Refer to your Motif documentation for a full explanation of resources and resource files.

Location of Resource Files

A resource database is constructed from several sources, listed here in descending order of precedence:

1. Command line options have the highest level of precedence and thus override duplicate settings in any resource file. The `.Xdefaults` file in the user's home directory `.Xdefaults` resource settings supersede duplicate settings in all other resource files, and so can be used to set individual user preferences. If you change the `.Xdefaults` file while the Motif Window Manager is running, reload the resource file with this command:

   ```
   xrdb -load .Xdefaults
   ```

2. The resource file that is named by the application class name, in the directory specified by the environment variable `XAPPLRESDIR`. This file can contain the user's or site administrator's preferences.

3. An application-specific resource file in the user's home directory. The user can override the global setting here.

4. The application-specific resource file that is named by the application class name, in the client directory `/usr/lib/X11/app-defaults`. These resources are then global to all users of a particular application.

Resources settings at each level override duplicate settings in lower-level resource files.

Colors

Panther running under a GUI offers access to many more color choices than character-mode. You can use Motif resource files to map Panther colors to Motif colors. These colors can be used in widget and screen properties.

Basic Color Mappings

Panther defines sixteen basic colors—eight highlighted and eight unhighlighted. These colors are displayed on the editor's color palette (refer to "How to Set an Object's Color with the Color Palette" on page 11-4 in *Using the Editors*):

The resource file can map basic Panther colors to any of the colors supported by Motif with this syntax:

`Prolifics.`*`prolColor: motifColor`*

`prolColor`
: One of the basic Panther colors listed earlier.

`motifColor`
: Set to one of the following:

  - A color name that appears in the `rgb.txt` file on your system. For example:

    ```
    Prolifics.blue: DarkSlateBlue
    ```

  - A hexadecimal RGB value. Hex specifications must be preceded by a # symbol. For example:

    ```
    Prolifics.green: #00a800
    ```

  Refer to the Motif *User's Guide* for details.

Note: To create custom color aliases beyond the sixteen basic colors, use the configuration map file—refer to "Configuration Map File" on page 45-25 in *Application Development Guide*.

Overriding Colors Set Within Panther

You can use Motif resources to set an application's default background and foreground colors and the colors of individual widget types. You can do so provided the following conditions are true:

- Applicable schemes in the configuration map file are set to `GUI` (refer to "Configuration Map File" on page 45-25 in *Application Development Guide*).
- The screen or widgets to be set from Motif resources have their Color Type properties set to Scheme.

How to Set an Application's Background and Foreground Colors

- Use Motif resources to set foreground and background resources from either the command line or in a resource file. When you invoke Panther, you can set color resources with `-bg` and `-fg` switches. For example:

  ```
  prodev -bg Yellow -fg Purple
  ```

  In a resource file, use this syntax:

  ```
  Prolifics*background:  motifColor
  Prolifics*foreground:  motifColor
  ```

  `motifColor` can be either a Motif color name or a hex value preceded by a # symbol.

How to Set Widget Colors

- Motif resource files can specify the color of widgets, and these settings can be used to override any color settings specified within Panther. For example, a foreground color setting for a text widget might look like this:

  ```
  Prolifics*XmText*foreground: blue
  ```

  This setting overrides any other foreground color for text widgets in applications of class `Prolifics`. A setting like the following changes the text widget for the specified screen `vidscreen`:

  ```
  Prolifics*vidscreen*XmText*foreground: blue
  ```

Motif Colors

Motif colors are listed in the `rgb.txt` file, often found in the directory `/usr/lib/X11`. If your `rgb.txt` file is located in a different directory, use the `Prolifics.rgbFilename`:directory resource to point to it. The `rgb.txt` file lists color names along with their red, green, and blue components. The colors are system dependent. Some common color names are listed in Table 4-1.

Table 4-1 Motif colors (partial listing)

alice blue	deep sky blue	light sky blue	papaya whip
antique white	dim gray	light slate blue	peach puff

Table 4-1 Motif colors (partial listing) ***(Continued)***

aquamarine	dim grey	light slate gray	peru
azure	dodger blue	light slate grey	pink
beige	firebrick	light steel blue	plum
bisque	floral white	light yellow	powder blue
black	forest green	lime green	purple
blanched almond	gainsboro	linen	red
blue	ghost white	magenta	rosy brown
blue violet	gold	maroon	royal blue
brown	goldenrod	medium blue	saddle brown
burlywood	gray	medium orchid	salmon
cadet blue	green	medium purple	sandy brown
chartreuse	green yellow	medium sea green	sea green
chocolate	grey	medium slate blue	sienna
coral	honeydew	medium turquoise	sky blue
cornflower blue	hot pink	medium violet red	slate blue
cornsilk	indian red	midnight blue	slate gray
cyan	ivory	mint cream	slate grey
dark goldenrod	khaki	misty rose	snow
dark green	lavender	moccasin	spring green
dark khaki	lavender blush	navajo white	steel blue
dark olive green	lawn green	navy	tan
dark orange	lemon chiffon	navy blue	thistle
dark orchid	light blue	old lace	tomato
dark salmon	light coral	olive drab	turquoise

Table 4-1 Motif colors (partial listing) ***(Continued)***

dark sea green	light cyan	orange	violet
dark slate blue	light goldenrod	orange red	violet red
dark slate gray	light gray	orchid	wheat
dark slate grey	light grey	pale goldenrod	white
dark turquoise	light pink	pale green	white smoke
dark violet	light salmon	pale turquoise	yellow
deep pink	light sea green	pale violet red	yellow green

Resource Options

This section describes resources that control behavior and the appearance of Panther running under Motif.

`baseWindow`

Controls whether a base window appears on the display. The base window is a special window that contains only a menu bar and a status line. Set this resource to `true` or `false`:

- true (default)—A base window appears on the display.
- false—No base window appears on the display. Any menu bar or status line that would appear in this window is lost. Refer to `formStatus` and `formMenus` to determine which status line and menu bars appear in the base window.

`focusAutoRaise`

Brings a screen to the top of the display when it gets the focus, with this setting:

```
Prolifics*focusAutoRaise: true
```

`fontList`

Sets the default font for the Panther application. For example:

```
Prolifics*fontList: fixed
```

Use a more fully qualified string to set fonts for parts of your application. For example, the following entry sets tooltip text to 18 point Helvetica:

```
Prolifics*toolbar*tooltip.fontList: *-helvetica-*-18-*
```

`formMenus`

Controls whether individual screens or windows have their own menu bars. `formMenus` can be set to true or false:

- false (default)—Only the base window displays a menu bar. Individual screens display no menu bar. Menu bars of all scopes, including screen-level, appear in the base window. If `baseWindow` is also false, then no menu bars appear at all.
- true—Individual screens display their own menu bar. Screens display menu bars of the scope `MNS_SCREEN` (screen-level). Only the active screen's menu bar is updated and active. Menu bars on inactive screens are inactive.

The base window, if there is one, displays menu bars of the scope `MNS_APPLIC` (application-level). The `SFTS` logical key can toggle between having the application-level or system-level menu bar displayed in the base window. If there is no base window, then no system- or application-level menu bars are displayed.

`formStatus`

Controls where status messages appear (not error messages). Error messages appear in dialog boxes, while status messages appear on the status line. This resource controls whether status messages appear on the base window's status line (the default), or on the active screen's (or window's) status line. The existence of the base window is controlled by the `baseWindow` resource.

There are five levels of status messages:

1. `d_msg_line`
2. `wait`
3. `field`
4. `ready`

5. `background`

   Background status messages can only appear in the base window. You can set `formStatus` to true or false:

   - `false` (default)—All status messages appear in the base window. Individual screens have no status line of their own. If there is no base window—that is, `baseWindow`: `false`—then there is no status line at all.
   - `true`—Background status messages appear in the base window. All other status messages appear in a status line on the active screen. The status line on individual screens appears at the bottom of the screen. Only the active screen's status line is updated. If a screen is not active, then its status line is not updated.

`introPixmap`

Specifies the image that appears during application startup. Refer to "Displaying a Picture on Widgets" on page 21-10 in *Using the Editors* for supported image file types. The specified image file name can include an explicit path; refer to "Storing the Image Files" on page 21-12 in *Using the Editors* for a description of the search algorithm Panther uses when you omit a path.

`labelString`

Sets the label for the specified edit item—for example, `edit_cut` or `edit_paste`. This statement sets the label for `edit_cut` to `Cut`:

```
Prolifics*XmMenuShell*edit_cut.labelString: Cut
```

`mnemonic`

Sets the mnemonic for keyboard access to the specified edit item—for example, `edit_cut` or `edit_paste`. This statement sets for `edit_cut's` mnemonic to `t`:

```
Prolifics*XmMenuShell*edit_cut.mnemonic: t
```

`moveThreshold`

Sets the number of pixels that the cursor must be moved to consider the movement to be a drag instead of a click. This applies when dragging out a rubber band when creating an object, or when trying to move or resize an object. If you want to move or resize a field by less than this amount, drag the mouse a larger distance than this resource is set to, then drag it back to the desired position without releasing the mouse button during the action. This resource defaults to 0.

`positionIsFrame`

When placing a window at a specific position on the display, the requested position can be for the placement of the frame or for the placement of the client window inside the frame. If the position is for the frame, set this resource to true (the default setting).

The window manager can have a resource of the same name. The value of the Panther resource should match the value of the window manager.

`pressAndMove`

Determines whether an object must be selected before it can be moved. When this resource is set to true (the default behavior), pressing on a widget selects it and allows it to be immediately dragged. When this resource is set to false, the widget must first be clicked on (press and release the mouse), and then pressed on again in order to move it.

Combined Resource Settings

You can combine resource settings in order to ensure compatibility with Windows and compliance with Motif standards, with respect to the appearance and behavior of menu bars and status lines:

- For compatibility with Panther running under Windows and backward compatibility with controlled release versions of Panther running under Motif, use the following default settings:

```
Prolifics*baseWindow:  true
Prolifics*formStatus:  false
Prolifics*formMenus:   false
```

- For full functionality with menu bars and status lines local to screens:

```
Prolifics*baseWindow:  true
Prolifics*formStatus:  true
Prolifics*formMenus:   true
```

- If you wish to have no base window:

```
Prolifics*baseWindow:  false
Prolifics*formStatus:  true
Prolifics*formMenus:   true
```

Do not use application level menu bars or background status messages with this combination; they will not appear.

Restricted Resources

The following items in the distributed Panther file must not be changed:

```
Prolifics*...*translations
Prolifics*keyboardFocusPolicy
Prolifics*...*traversalOn
```

You can change all other items, including `Mwm*Prolifics*keyboardFocusPolicy`.

Global Resource and Command Line Options

The resources in Table 4-2 are global settings that function on an application-wide basis. You can also specify them on the command line, as you can standard X Toolkit command line options. Refer to the *X Toolkit* manual for a full list of command line switches.

Table 4-2 Global resource options

Resource	Type	Command Line	Description
foreground	string	-bg color	Sets unhighlighted white foregrounds to color.
background	string	-fg color	Sets unhighlighted black backgrounds to color.
ownColormap	boolean	-cmap (on) +cmap (off) D	Tells Panther whether to use its own color map. Use `on' for systems with limited colors.
D = default			

The following illustrates a sample command line resource setting:

```
prodev -fg 'white' myscreen.scr
```

Widget Hierarchy

Widgets are arranged in a parent-child hierarchy. The tables in this section describe the widget hierarchy in Panther running under Motif. This information is required in order to set resources for particular widgets or classes of widgets in your application. Refer to the `OSF/Motif Programmer's Guide` for more information on widgets, widget classes, and the resources associated with them.

Base Screen

The base screen in a Panther application is an `ApplicationShell` widget. Its class is given by the first argument to the initialization routine, and its name is the name of the application program (the value of `argv[0]` in `main`). If the `baseWindow` resource is set to `false`, then this shell is created but never displayed.

Note: Avoid application program names that contain periods or asterisks; the resource parser interprets these as special characters. Periods that precede widget name extensions are exempt from this restriction: extensions are stripped from Panther names before Motif uses them.

By default, Panther has class name `Prolifics` and application name `prodev` or `prorun`.

Table 4-3 lists the widget hierarchy for the base screen.

Table 4-3 Base screen widget hierarchy

Name	Widget Class
application-name	ApplicationShell... (given by initialization routine)
`main`	`XmMainWindow`
`statusForm`	`XmForm`
`statusText`	`XmText`
`menubar`	`XmRowColumn`
`toolbar`	`XmRowColumn`

The `status` area is used for the Panther status line in the base screen.

Dialog Boxes

The creation of dialog boxes is handled by Panther in some cases, and by Motif in others. Table 4-4 lists the appropriate function or Motif resource associated with the creation of specific dialog box types:

Table 4-4 Dialog box types

Dialog box type	Created by
File selection	`sm_filebox` library function
Message	Need to post message
Error message	`XmCreateErrorDialog`
Query message	`XmCreateQuestionDialog`

Panther specifies the message string, which buttons appear, and which button is the default. The Panther message call can specify the icon to appear. Other options, like the title bar text, can be set in the resource file.

Children of dialog boxes are handled by Motif. Refer to your Motif documentation for details.

Panther Screens

The widgets used in Panther screens are all subclasses of the Motif `shell` widget. The `shell`'s parent is the `ApplicationShell`. Table 4-5 lists the widget hierarchy for Panther screens.

Table 4-5 Widget hierarchy for Panther screens

Name	Widget Class
screen-name	`...TopLevelShell`
`message_popup`	`XmDialogShell`
`message`	`XmMessageBox...`
`filebox_popup`	`XmDialogShell`
`fileBox`	`XmFileSelectionBox...`
`scroll`	`XmMainWindow`
`clip`	`SmSimpleManager`
`area`	`SmSimpleManager`
`statusForm`	`XmForm`
`statusText`	`XmText`
`scrollbar`	`XmScrollBar`
`scrollbar`	`XmScrollBar`
`menubar`	`XmRowColumn`
`toolbar`	`XmRowColumn`

Panther screens have a status line only if the value of the `formStatus` resource is `true`. They have a menu bar only if `formMenus` is `true`.

New screens are named `shell` before they are saved.

Because the name of the shell used for Panther screens is the screen name, resources can be restricted to a specific screen if you begin the specification with `class*screen_name`. For example, `Prolifics*vidscrn...` begins a specification for a screen named `vidscrn` in an application of class `Prolifics`. Resources restricted to a named screen are equivalent to screen property. For example:

```
Prolifics*vidscrn.background: gold
```

is the same as specifying a screen color property.

`area` is the parent widget for all the widgets on a Panther screen. To place your own widgets on a Panther screen, you need the widget ID of `area`. The library function `sm_drawingarea` returns the widget ID of `area`. A related function, `sm_translatecoords`, translates Panther screen coordinates into pixel coordinates relative to the upper left hand corner of `area`.

Widgets

Panther widgets are created as child widgets of `area`. If a widget has a name, its corresponding Motif widget gets the same name. If a field doesn't have a name, its Motif widget is named `_fld#`, where # is the field number. In a named array consisting of multiple fields, each widget has the same name. Motif widgets that represent multiple fields take the name of the first field.

Motif variants of `sm_widget` such as `sm_xm_widget` return a widget ID. Asterisks in the table below indicate which widget is returned by `sm_widget` in cases where there is more than one possibility. If the widget returned by `sm_widget` is not the one you are looking for, use `XtParent` to obtain the widget ID of its parent. This is particularly useful when working with scale widgets and scrolling multiline and list box widgets.

Table 4-6 lists the widget hierarchy for Panther widgets. Some entries in the table have prefixes or suffixes with their names. For example, `field-nameSW` indicates that the widget's name is composed of the field's name followed by the characters `SW`.

Table 4-6 Widget hierarchy for Panther widgets

Object	Name	Widget Class
box	`box`	`...XmFrame*`
	`title`	`XmLabel`
line	`separator`	`...XmSeparator`

Table 4-6 Widget hierarchy for Panther widgets *(Continued)*

Object	Name	Widget Class
graph	box	...XmDrawingArea
grid	*grid-name*	...SmMatrix*
	horizScroll	XmScrollBar
	vertScroll	XmScrollBar
	clip	SmClip
	textField	XmText
single line text	*field-name*	...XmText
static label	*field-name*	...XmLabel
radio button	*field-name*	...XmToggleButton
toggle button	*field-name*	...XmToggleButton
check box	*field-name*	...XmToggleButton
dynamic label	*field-name*	...XmLabel
push button	*field-name*	...XmPushButton
combo box	*field-name*	...XmForm
	text	XmText*
	arrow	XmArrowButton
	*field-name*_pane	XmRowColumn
	label-text *label-text* ... *label-text*	XmPushButton XmPushButton ... XmPushButton
multiline text	*field-name*	...XmText
multiline text with scroll bars	*field-name*SW	...XmScrolledText
	field-name	XmText*

Table 4-6 Widget hierarchy for Panther widgets *(Continued)*

Object	Name	Widget Class
list box	*field-name*	...XmList
list box with scroll bars	*field-name*SW	...XmScrolledList
	field-name	XmList*
option menu	*field-name*	...XmRowColumn*
	popup_*field-name*_pane	...XmMenuShell
	*field-name*_pane	...XmRowColumn
	label-text *label-text* ... *label-text*	XmPushButton XmPushButton ... XmPushButton
scale	*field-name*	...XmScale
	scale_scrollbar	XmScrollBar*

To refer to a whole class of widgets, use the widget class. For example, `Prolifics*XmText` refers to all text widgets. To refer to a class of widgets on a screen, use the screen name followed by the widget class. For example, `Prolifics*empscreen*XmText` refers only to text widgets on the screen `empscreen`. To refer to an individual widget, use the screen name followed by the widget's name. For example, `Prolifics*empscreen*empname` refers only to the `empname` widget on the screen `empscrn`.

In the option menu widget, the text field and the popup pane are linked through the `subMenuID` field of the `RowColumn` widget. Because the push buttons in the option menu are named by their contents, it is easier to set a resource for all the push buttons in an option menu than it is to set a resource for an individual button.

Menus and Toolbars

Menus—instantiated as menu bars and their submenus, as popup menus, or as toolbars—are created within `RowColumn` widgets. Menu bars are children of either the base screen's or an individual screen's `MainWindow`. Submenus are children of

MenuShells, but the name of the shell is unclear, because Motif reuses these shells. If a new shell is created, its name is popup_*submenu-name*. Specify resources for a submenu as follows:

Prolifics*XmMenuShell.*submenu-name*

Table 4-7 lists the hierarchy for menus and popup menus.

Table 4-7 Hierarchy for menus and popup menus

Object	Name	Widget Class
menu bar	*menu-name*	...XmRowColumn...
submenu	(name varies)	...XmMenuShell
	submenu-name	XmRowColumn...
popup menu	*application-name*	ApplicationShell
	dummy	TransientShell
	popup_popupmenu	XmMenuShell
	popupmenu	XmRowColumn...
toolbar	toolbar	...XmRowColumn

Submenus pop up through the auspices of a CascadeButton widget. A submenu is tied to its CascadeButton via the XmNsubMenuID field of the button.

Menu items are children of the menu's RowColumn widget. Table 4-8 lists their hierarchy.

Table 4-8 Hierarchy for menu item types

Menu script keyword	Name	Widget class
ACTION	*label-text*	...XmPushButton
EDCLEAR	*edit_clear*	...XmPushButton
EDCOPY	*edit_copy*	...XmPushButton
EDCUT	*edit_cut*	...XmPushButton

Table 4-8 Hierarchy for menu item types *(Continued)*

Menu script keyword	Name	Widget class
EDDEL	*edit_delete*	...XmPushButton
EDPASTE	*edit_paste*	...XmPushButton
EDSELECT	*edit_select*	...XmPushButton
SUBMENU	*label-text*	...XmCascadeButton
SEPARATOR	separator	...XmSeparator
TOGGLE	*label-text*	...XmPushButton
WINLIST	*window-name* *window-name* ... *window-name*	...XmPushButton ...XmPushButton ... ...XmPushButton
WINOP	*windows_raise*	...XmPushButton

Toolbar items are children of the menu's RowColumn widget. Table 4-9 lists their hierarchy.

Table 4-9 Hierarchy for toolbar components

Object	Name	Widget class
button	*label-text*	...XmPushButton
separator	*_tool#*	...XmDrawingArea

Sample Motif Resource File

```
!#############################################################
!### Geometry Resources                                    ###
!#############################################################

! Set this to true if you use interactive placement for your
! window manager and want Prolifics to respect it. Default
! value is false if not specified.

Prolifics.interactivePlacement: false

! Set Geometry for the base window.

! Default

Prolifics.geometry:        635+0+0

! Use this as an example if Interactive Placement is set to
! true.

!Prolifics.geometry:       635

!#############################################################
!### Double Click Resources                                ###
!#############################################################

! Set your preference for double click time in thousandths of
! a second (e.g. 500 = 1/2 second). If you find that double
! clicks are being treated as separate clicks, then raise
! this value. If separate clicks are unexpectedly being
! treated as a double click, then lower this value.

Prolifics.multiClickTime:   400

!#############################################################
!### Color Resources                                       ###
!#############################################################

! Set the location of the X rgb color database file.
! This must be changed to reflect the site-specific location.

Prolifics.rgbFileName:/usr/lib/X11/rgb.txt

! Set any GUI colors in your color scheme.
```

```
Prolifics*foreground:white

Prolifics*background:dark slate gray

! Set the 16 Prolifics colors. These are mandatory, as
! Prolifics needs a way to map the 16 Basic colors to actual
! GUI values. But you can modify the values to suit your
! needs.

Prolifics.black:#000000
Prolifics.blue:#0000a8
Prolifics.green:#00a800
Prolifics.cyan:#00a8a8
Prolifics.red:#a80000
Prolifics.magenta:#a800a8
Prolifics.yellow:#a85400
Prolifics.white:#a8a8a8
Prolifics.hi_black:#545454
Prolifics.hi_blue:#5454ff
Prolifics.hi_green:#54ff54
Prolifics.hi_cyan:#54ffff
Prolifics.hi_red:#ff5454
Prolifics.hi_magenta:#ff54ff
Prolifics.hi_yellow:#ffff54
Prolifics.hi_white:#ffffff

!################################################################
!### Screen Editor Resources                                  ###
!################################################################

! Resources in this section only affect the operation of the
! Prolifics Screen Editor. They should be removed when
! supplying a resource file to your end-users.

! Determines how many pixels the mouse pointer must move in
! Edit Mode before being considered a drag.

Prolifics.moveThreshold:             6

! Set the path of the Editor help file.  This must be set in
! order to use online Help within the Screen Editor.

! Prolifics.editorHelpPath:          $SMBASE/docs/books

! Splits the Property Window drop down into two or more
! columns if more than 15 items appear, to ensure that all of
! the list is visible. Users of high-resolution monitors may
! use a higher threshhold as the value of numColumns: 30 on a
! 17" 1024x768 monitor, for instance.

Prolifics*smpropty*proptext_pane*packing:      PACK_COLUMN
```

```
Prolifics*smpropty*proptext_pane*orientation:  HORIZONTAL

Prolifics*smpropty*proptext_pane*numColumns:   15

!#################################################################
!### General Resources                                         ###
!#################################################################

! Point this resource to an XPM, GIF or JPEG file to display
! when the application starts up.

Prolifics.introPixmap:

! Set this resource to display the application name on the
! base window ! title bar.

Prolifics.title:Prolifics for Motif

! Set this to false to remove all the extra insert cursors
! that Motif 1.2 displays by default. Defaults to true if not
! specified.

Prolifics*cursorPositionVisible:false

! Uncomment the following line to enable Tear-Off Menus.
! Tear-Off Menus are disabled by default.

!Prolifics*tearOffModel:TEAR_OFF_ENABLED

! Set the following resource to true to force all Prolifics
! stacked windows to be application modal dialogs. Defaults
! to false.

Prolifics*stackedWindowsAreDialogs:false

! Specify the application-wide default font. Use a more fully
! qualified string to set fonts for parts of your application
! (e.g. Prolfiics*message*fontlist for message boxes).

Prolifics*fontList:fixed

! Example of specifying the font for the tooltip text.

!Prolifics*toolbar*tooltip.fontList:fixed

! The following prevents file selection boxes from changing
! their size when the Filter button is pressed. This may be
! changed according to taste.Defaults to RESIZE_ANY if not
! specified.

Prolifics*XmFileSelectionBox.resizePolicy:RESIZE_NONE

! Newer filebox versions have a separate filter field.
```

```
Prolifics*XmFileSelectionBox.pathMode:PATH_MODE_RELATIVE

! Set the full path and name of the runtime help file. This
! option contains the name of the help file used by the
! external help system you've installed.  The default value
! is empty.

Prolifics.helpFile:

! Under VMS, text widgets seem to grab the selection
! unless the following is set.

Prolifics*area*navigationType:NONE

!#################################################################
!### Drag and Drop                                             ###
!#################################################################

! The Motif Drag and Drop protocol is often buggy in many
! commercial Motifs. In addition, it increases widget
! creation overhead and may affect performance. These
! resources are set to disable Motif Drag and Drop. This does
! NOT affect the ability to drag and drop widgets within the
! Prolifics Screen Editor. You may enable these resources by
! changing their values.

Prolifics*dragReceiverProtocolStyle:DRAG_NONE

Prolifics*dragInitiatorProtocolStyle:DRAG_NONE

Prolifics*dropSiteActivity:DROP_SITE_INACTIVE

!#################################################################
!### Edit and Windows Menu Resources                           ###
!#################################################################



! Text and mnemonics to be used in window and edit menu
! items.

Prolifics*XmMenuShell*windows_raise.labelString:Raise All
Prolifics*XmMenuShell*windows_raise.mnemonic:R
Prolifics*XmMenuShell*edit_cut.labelString:Cut
Prolifics*XmMenuShell*edit_cut.mnemonic:t
Prolifics*XmMenuShell*edit_copy.labelString:Copy
Prolifics*XmMenuShell*edit_copy.mnemonic:C
Prolifics*XmMenuShell*edit_paste.labelString:Paste
Prolifics*XmMenuShell*edit_paste.mnemonic:P
Prolifics*XmMenuShell*edit_delete.labelString:Delete
Prolifics*XmMenuShell*edit_delete.mnemonic:D
```

```
Prolifics*XmMenuShell*edit_select.labelString:Select All
Prolifics*XmMenuShell*edit_select.mnemonic:S
Prolifics*XmMenuShell*edit_clear.labelString:Clear
Prolifics*XmMenuShell*edit_clear.mnemonic:l

! The standard Prolifics key file for X, "xwinkeys", maps
! unmodified, shifted, and control function keys 1-12 into
! the Prolifics logical keysPF1-12, SPF1-12, and SFT1-12.
! This conforms to the standard key conventions used for
! Prolifics on character terminals.

! Unfortunately, these may conflict with the fallback or
! vendor-specific default bindings which Motif uses for its
! virtual keysyms. The following line disables all of the
! virtual keysyms within a Prolifics application. (Actually,
! the default binding for osfMenuBar is remapped to F25.
! If we were to unmap it, the Motif library would reset it
! to F10.)

! If you prefer the standard Motif usage for the function
! keys, you can change the Prolifics key file to avoid
! the keys which conflict with Motif.
! The following line can then be commented-out.
! If you retain any of the following, you must retain
! entries for both osfMenu and osfMenuBar or the
! program will crash on some versions of Motif.
! (You can change to which keys they are bound.)

Prolifics*defaultVirtualBindings:       \n\
       osfMenu:        <Key>F26         \n\
       osfMenuBar:     <Key>F25         \n\
       osfActivate:    <Key>KP_Enter    \n\
       osfCancel:      <Key>Escape      \n\
       osfDown:        <Key>Down        \n\
       osfLeft:        <Key>Left        \n\
       osfRight:       <Key>Right       \n\
       osfUp:          <Key>Up          \n\
       osfBackSpace:   <Key>BackSpace   \n\
       osfDelete:      <Key>Delete      \n\
       osfHelp:         <Key>F1

!
! Sun CDE patches
!

Prolifics*enableToggleColor:            false
Prolifics*enableCDEColorFactors:        false
Prolifics*enableToggleVisual:           false
Prolifics*enableEtchedInMenu:          false
```

CHAPTER

5 Character Mode Settings

A number of behavior variables are only applicable to running Panther in character-mode. These variables include those that are used to control label text display and display of the active screen.

For more information about behavior variables in general, refer to Chapter 5, “Character Mode Settings.”

For information about setting display characteristics for a given character-mode terminal type, refer to Chapter 7, “Video File.”

Label Text Display

Two types of variables control the display attributes of label text in character-mode applications:

- Variables that set display of label mnemonics: `AC_KEEPATTRS`, `AC_SETATTRS`, and `AC_SWATTRS`. These variables only affect widgets that have Label properties—for example, dynamic labels and push buttons.
- Variables that grey-out text in inactive widgets—that is, widgets that cannot get focus: `FE_KEEPATTRS`, `FE_SETATTRS`, and `FE_SWATTRS`. These variables only

affect menu items and widgets that have Label properties, such as dynamic labels and push buttons.

AC_KEEPATTRS
: Specifies those attributes to retain (AND) for keyboard mnemonic characters that are emphasized. The default includes all attributes currently assigned. Refer to Table 2-1 on page 2-4 for a list of the keywords.

AC_SETATTRS
: Sets display attributes to emphasize keyboard mnemonic characters. The default is none. Refer to Table 2-1 on page 2-4 for a list of the keywords.

AC_SWATTRS
: Sets display attributes that are switched for keyboard mnemonic characters. The default is HILIGHT WHITE for menu items HILIGHT CYAN. Refer to Table 2-1 on page 2-4 for a list of valid keywords.

FE_KEEPATTRS
: Specifies those attributes to retain (AND) for the label text of inactive items. The default includes all attributes currently assigned. Refer to Table 2-1 on page 2-4 for a list of the keywords.

FE_SETATTRS
: Sets display attributes for label text. of inactive items The default setting is none. Refer to Table 2-1 on page 2-4 for a list of the keywords.

FE_SWATTRS
: Sets display attributes that are toggled when graying is turned on and off. The default setting is HILIGHT. Refer to Table 2-1 on page 2-4 for a list of the keywords.

Screen Attributes

The EMPHASIS variable determines the style of emphasis used to indicate the current, or active, screen. The options are drop shadows, graying, or border highlights. Drop shadows appear to cast a shadow from the active screen over underlying screens. Graying changes the display attributes of all screens except the active one: highlights turn off and colors change to monochrome by default. You can set graying attributes

with video file entries EMPHASIS_KEEPATT and EMPHASIS_SETATT (page 7-33). Border highlighting turns on the highlight characteristic for the active screen border. An application that runs in character-mode can use any one style, or drop shadows and graying can be used together.

EMPHASIS
: Specifies emphasis style:

  DROPSHADOW
  : Draws a shadow at the screen's uppermost right and bottom edges. The shadow is two columns wide and one line deep. The right shadow starts one space below the screen's upper edge, while the bottom shadow starts two columns from the screen's left edge. The bottom shadow is indented two spaces from the left edge of the screen. The shadow is formed by graying the underlying text.

  GRAYBKGD
  : Grays background screens. Only the active screen retains its original display attributes.

  HIBORDER
  : Highlights the border of the active screen.

  NONE
  : Disables display emphasis. (D)

Menus

These variables control the way menus appear and behave in character-mode applications. You can change these variables at runtime with sm_option.

MB_KEEPATTRS
: Specifies the attributes to be retained (Handed) for keyboard mnemonic characters that are emphasized. The default includes all attributes currently assigned. Refer to Table 2-1 on page 2-4 for a list of the keywords.

MB_SETATTRS
: Specifies display attributes to emphasize keyboard mnemonic characters. The default is none. Refer to Table 2-1 on page 2-4 for a list of the keywords.

MB_SWATTRS
: Specifies display attributes that are switched for keyboard mnemonic characters. The default is HILIGHT WHITE for menu items HILIGHT CYAN. Refer to Table 2-1 on page 2-4 for a list of valid keywords.

MB_BORDSTYLE
: Specifies border style:

  NOBORDER
  : No border.

  style number
  : A number between 0 and 9 indicates a style. Default is 1.

MB_BORDATT
: Specifies border display attributes for menu bars. The default is MB_BORDATT = B_WHITE BLACK. Refer to Table 2-1 on page 2-4 for a list of the keywords.

MB_DISPATT
: Set display attributes for text in menu bars. The default is MB_DISPATT = B_WHITE BLACK. Refer to Table 2-1 on page 2-4 for a list of the keywords.

MB_FLDATT
: Set display attributes for unselected menu bar options. The default is MB_FLDATT = B_WHITE BLACK. Refer to Table 2-1 on page 2-4 for a list of the keywords.

MB_HBUTDIST
: Set the distance between menu items on a horizontal menu. Default is 2 columns.

MB_LINES_PROT
: Set the number of top lines reserved for a menu bar. Default is 1.

MB_SYSTEM
: Specifies presence of system menu on menu bars:

  OK_SYSTEM
  : System menu item (==) appears on menu bar. (D)

  NO_SYSTEM
  : System menu item does not appear on menu bar.

Miscellaneous

`IN_BLOCK`
: Sets cursor appearance:

  `OK_NOBLOCK (d)`
  : Cursor occupies one character position in a field.

  `OK_BLOCK`
  : Current field is changed to reverse video to simulate a large cursor. The cursor occupies the entire field.

`MB_ACCATT`
: Sets the attributes for mnemonic characters on a menu bar. The default is `HILIGHT | WHITE`

`ZW_BORDSTYLE`
: Sets border style for zoom windows:

  `NOBORDER`
  : No border.

  `style number`
  : A number between 0 and 9 indicates a style. Default is 1.

`ZW_BORDATT`
: Sets border attributes for zoom windows. The default value `RED HILIGHT B_CONTAINER`. Refer to Table 2-1 on page 2-4 for a list of the keywords.

`SMLPRINT`
: Specifies the operating system command that is invoked through the local print key (LP). Use this variable to issue a print command that prints the current screen by substituting `%s` for the file name. For example, the print screen command might look like this:

  ```
  SMLPRINT = lpr %s
  ```

  This variable can be defined in a setup file, overridden by setting it at the system environment, and changed at runtime with `sm_soption`.

CHAPTER

6 Key Translation File

A Panther application must allow users to enter ASCII data characters and to indicate certain logical key values to Panther—`EXIT`, `XMIT`, `PF1`, `SPF1`, and so on. Because physical keyboards vary from system to system, Panther uses a key translation file to translate sequences that users enter on their physical keyboards into logical keys that Panther understands.

The Panther distribution provides key translation files (in the `config` directory), that support more than 100 terminal types. One of the distributed key files should work for you. However, you might want to create, modify, or examine the key translation file if you:

- Have a unique keyboard that does not have extended keys corresponding to the Panther logical keys (also referred to as Panther logical keys).
- Want to change or define key assignments for specific logical keys.
- Want to assign or change key labels for logical keys.

This chapter describes:

- What a key translation file does
- How to read the entries in the key translation file (page 6-5)
- How to view key sequences produced by your keyboard with the `showkey` utility (page 6-4)
- How to convert key translation files to binary format with the `key2bin` utility
- Using alternate key translation files in portable applications (page 6-16)

The Role of the Key Translation File

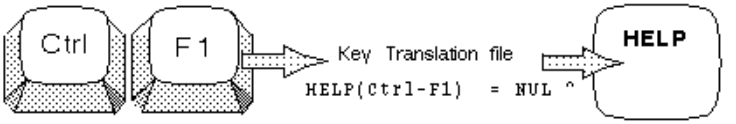
During initialization, Panther reads the key translation file specified by SMKEY (in the smvars file) into memory. The value for SMKEY is determined by the terminal type indicated in your SMTERM or system TERM variable. Refer to Chapter 2, "Application Variables," for more information about setting the SMKEY variable. Now Panther can recognize how your physical keys map to the logical keys.

Panther logical keys are defined as hexadecimal values in the include file smkeys.h. This file is terminal-independent, while key translation files are terminal-dependent. ibmkeys, vt220keys, wy75keys are a few of the available key translation files.

For the most part, ASCII data key, logical values between 1 and hex FF, require no translation and are not included in the key translation file. However, some keyboards do not have the same number of function keys, and most do not have logical keys named HELP or XMIT (Transmit). Therefore, the key translation file specifies a physical key which transmits a unique code or sequence of codes which, in turn, works as a HELP key. Another key works as the XMIT key, and so on. Logical values between hex 100 and hex 1FF are cursor control and editing keys; values greater than hex 1FF are function keys.

So, when you press a key, the keyboard generates either a single ASCII data character, or a sequence of characters beginning with an ASCII control code. Panther converts these characters into logical keys, numbered between 1 and 65535 inclusively, before processing them—hence determining if a data character or control character was received.

If the key input is a control character, Panther searches the key translation file for a sequence beginning with that character. If there is a match, additional characters are read until the entire sequence is found, and returns the logical value. So, Panther is able to interpret XMIT when you press a Do key, or an End key, or whatever physical key is mapped to XMIT in your key translation file.



If a match is found on the initial character, but not the whole sequence, the entire sequence is discarded. If there is no match at all with the entered control character, it is returned unchanged; this is useful for machines such as IBM PCs that use control codes for displayable characters. (For more information about key translation, refer to "Processing Keyboard Input" on page B-2 in *Upgrade Guide*.)

With a one-to-one mapping there are not enough keys on a commercially available keyboard to represent the entire Panther logical keyboard—and for your application, you may not need to use or map all the logical keys. However, to accommodate its larger logical keyboard, Panther combines two or more physical keys to represent a logical key. For example, on a PC the logical key `ZOOM` is often mapped to Alt-Z.

Learn Your Key Mapping

You can print out the ASCII key translation file that supports your keyboard. Key translation file names begin with a mnemonic for the type of terminal you are using, and end with `keys`. For example, `vt100keys` is the key translation file for a VT100 terminal. All key translation files distributed with Panther adhere to this convention.

If you use Panther on different terminals with varying operating systems and keyboards, then each terminal must have its own key translation file. If necessary, you can create more than one key translation file for a terminal. In this way you can tailor the key mapping for a particular application.

Refer to "Processing Keyboard Input" on page B-2 in *Upgrade Guide* for a discussion of key processing in Panther applications.

Viewing Key Sequences

As an aid to editing key files, the `showkey` utility is provided. The `showkey` utility prints out the key sequence of keys that you enter. To run this utility, type `showkey` at a system prompt. (This is a Panther executable, and as such, the variables `SMVARS` and `SMBASE` must be appropriately defined to run this utility.) The Showkey Utility window appears:

In order to assure that you are not mistyping your keystrokes, showkey expects you to enter your keystrokes twice. If you do not press the same key sequence twice, it will not display the generated characters. Type `x` twice in a monospace font to exit the `showkey` utility.

If you want to bind the keystroke(s) to a Panther logical key, copy the decoded keystroke text to your key translation file. For example if you wanted to assign `Shift-F10` on your keyboard to the logical key which starts the debugger, you would:

1. Start the showkey utility.
2. Enter `Shift-F10` twice.

The showkey window displays the character sequence that the keystroke transmits. On one platform, the sequence for `Shift-F10` was `ESC [ 4 5 ~`

3. Bind the key label and character sequence to the debugger hot key:

   ```
   DBUG(Shift-F10)= Esc [ 4 5 ~
   ```

4. Run the `key2bin` utility on your altered key file.

5. Type `xx` to exit the utility.

Key Translation File Syntax

Each entry in a key translation file has the format:

`logical-key(key-label) = character-sequence`

Lines beginning with a pound sign # are treated as comments. They are ignored by the `key2bin` utility.

`logical-key`

The mnemonic or the hexadecimal value of a Panther logical key. For example, the logical `TRANSMIT` key is represented by the mnemonic `XMIT` or by the hex value `0x104`. The mnemonics and hex values for all logical keys are defined in `smkeys.h`. Refer to Table 6-1 for a list of these mnemonics and values.

You can assign the same *`logical-key`* to two (or more) different *`character-sequences`*. For example, in a key translation file for the PC, you can make these entries:

```
HELP(Ctrl-F1) = NUL ^
HELP(Ctrl-Home) = NUL w
```

When these entries are used with a PC, both `Ctrl-F1` and `Ctrl-Home` will execute `HELP`.

`key-label`

The letter, numeral, character, or character string engraved on a physical keytop. One or more physical key labels may be included inside the

parentheses, for example `(Alt-F1)`. *`key-label`* is optional. It can be accessed at runtime through various library functions and the `%K` escape in status line messages. (Refer to `sm_d_msg_line`.) Key labels are often helpful in user messages and prompts. If *`key-label`* is not specified, the logical key value is used in screen displays.

`character-sequence`

The sequence of characters, up to six characters not including blanks, produced by a keystroke. Panther translates *`character-sequence`* to be the logical key on the left of the equal sign.

When a physical key is pressed, it transmits a character code which is unique from the code produced by any other key on the keyboard. Each complete character sequence has only one logical value. Although a sequence may include another as a substring.

In character-mode, if you use key sequences that are lead-ins of other sequences, you must assign a timing interval with the video file entry keyword `KBD_DELAY`. For example, if you have defined one logical value to `ESC`, another to `ESC [ F` and have set `KBD_DELAY` to 5, Panther will wait a half of a second after `ESC` is pressed to resolve the ambiguity. If during that interval `[ F` is not received (or any other identifiable sequence that is prefixed with `ESC`) Panther passes on the logical value of `ESC`. For more information on keyboard delay, refer to page 7-24, “KBD_DELAY.”

Example

The following example is an excerpt from a key translation file:

```
EXIT (F1) = SOH @ CR
XMIT (Enter) = SOH O CR
TAB = HT
BACK = NUL SI
BKSP = BS

# These are the arrow keys
RARR = ESC [ C
LARR = ESC [ D
UARR = ESC [ A
DARR = ESC [ B

# The next entry uses a hex value rather
# than a mnemonic for logical-key
0x108 = DEL
```

Key Mnemonics and Hexadecimal Values

Table 6-1is derived from the include file `smkeys.h` which defines the Panther logical keyboard. Some entries are required by the editor and are indicated with double asterisks (**).

Table 6-1 Panther Logical Keyboard—key mnemonics and hexadecimal values

Logical key mnemonic	Hex value	Description
CLWIN	0x100	close the current window
SYSMN	0x101	access system menu on active screen/window
CLAPP	0x102	close the application
EXIT	0x103	exit**
XMIT	0x104	transmit**
HELP	0x105	help on field*
FHLP	0x106	help on screen or form
CSHLP	0x107	context help when widget is clicked in Help Mode
BKSP	0x108	backspace*
TAB	0x109	tab*
NL	0x10a	newline*
BACK	0x10b	backtab*
HOME	0x10c	go to first field on screen*
NCARD	0x10d	next tab card
DELE	0x10e	delete character*
INS	0x10f	insert/overwrite character toggle*
LP	0x110	local print (SMLPRINT)

Table 6-1 Panther Logical Keyboard—key mnemonics and hexadecimal values

Logical key mnemonic	Hex value	Description
FERA	0x111	clear field*
CLR	0x112	clear all unprotected*
SPGU	0x113	display next page of data (scroll down)
SPGD	0x114	display previous page of data (scroll up)
PCARD	0x115	previous tab card
LSHF	0x116	left shift
RSHF	0x117	right shift
LARR	0x118	left arrow*
RARR	0x119	right arrow*
DARR	0x11a	down arrow*
UARR	0x11b	up arrow*
LWRD	0x11c	left to previous word
RWRD	0x11d	right to next word
REFR	0x11e	refresh screen*
EMOH	0x11f	go to last field on screen
INSL	0x120	insert line*
DELL	0x121	delete line*
ZOOM	0x122	zoom on field*
SFTS	0x123	soft key select
MTGL	0x124	toggle menu mode
VWPT	0x125	viewport
MOUS	0x126	indicate mouse event
MNBR	0x127	access menu bar

Table 6-1 Panther Logical Keyboard—key mnemonics and hexadecimal values

Logical key mnemonic	Hex value	Description
CYCL	0x128	cycle through a set of sibling windows
DBUG	0x129	hot key for debugger
WMODE***	0x12a	toggle control code display
WTAB***	0x12b	hard tab
WNL***	0x12c	hard new line
ITSEL	0x12d	item selection screen key
BOLN	0x12e	beginning of line (widget)
EOLN	0x12f	end of line (widget)
MDBL	0x131	mouse double click
OPTDN	0x132	option menu drop down key
BOFD	0x133	beginning of entry field
EOFD	0x134	end of entry field
ADDM	0x135	add mode toggle in list box
EXT	0x136	extend selection to select contiguous list items
EXTD	0x137	extend selection with down arrow in text field or list box
EXTU	0x138	extend selection with up arrow in text field or list box
EXTL	0x139	extend selection with left arrow in text field
EXTR	0x13A	extend selection with right arrow in text field
EXTWL	0x13B	extend selection one word left in text field

Table 6-1 Panther Logical Keyboard—key mnemonics and hexadecimal values

Logical key mnemonic	Hex value	Description
EXTWR	0x13C	extend selection one word right in text field
EXTLB	0x13D	extend selection to start of line in text field
EXTLE	0x13E	extend selection to end of line in text field
EXTPU	0x13F	extend selection up one page in text field or list box
EXTPD	0x140	extend selection down one page in text field or list box
EXTFB	0x141	extend selection to start of field or list box
EXTFE	0x142	extend selection to end of field or list box
SLWRD	0x143	select current word
SLALL	0x144	select entire text field
ALSYS	0x83D	access and open system menu

** Recommended entries. **Entries required by* `prodev`*. ***Used in wordwrap fields.*

Note: The documentation on error messages and error acknowledgment often refers to an error acknowledgment key whose default is the space bar. Because the space bar is a data entry key, it cannot be used as a logical key. Instead, the key is defined as the application behavior variable `ER_ACK_KEY`. You can change the entry in the `SMVARS` file, in a setup file, in the system environment, or at runtime with the library function `sm_option`.

Table 6-2 includes the mnemonics and hexadecimal values for function keys (`PF`), shifted function keys (`SPF`), and `ALT` keys. Table 6-3 includes the mnemonics and hexadecimal values for application function keys (`APP`).

Table 6-2 Hexadecimal values for function and ALT keys

PF	Hex	SPF	Hex	ALT	Hex
PF1	0x6101	SPF1*	0x4101	ALTA	0x4103
PF2*	0x6201	SPF2*	0x4201	ALTB	0x4203
PF3*	0x6301	SPF3*	0x4301	ALTC	0x4303
PF4*	0x6401	SPF4*	0x4401	ALTD	0x4403
PF5*	0x6501	SPF5*	0x4501	ALTE	0x4503
PF6*	0x6601	SPF6*	0x4601	ALTF	0x4603
PF7*	0x6701	SPF7	0x4701	ALTG	0x4703
PF8*	0x6801	SPF8	0x4801	ALTH	0x4803
PF9*	0x6901	SPF9	0x4901	ALTI	0x4903
PF10*	0x6a01	SPF10	0x4a01	ALTJ	0x4a03
PF11	0x6b01	SPF11	0x4b01	ALTK	0x4b03
PF12	0x6c01	SPF12	0x4c01	ALTL	0x4c03
PF13	0x6d01	SPF13	0x4d01	ALTM	0x4d03
PF14	0x6e01	SPF14	0x4e01	ALTN	0x4e03
PF15	0x6f01	SPF15	0x4f01	ALTO	0x4f03
PF16	0x7001	SPF16	0x5001	ALTP	0x5003
PF17	0x7101	SPF17	0x5101	ALTQ	0x5103
PF18	0x7201	SPF18	0x5201	ALTR	0x5203
PF19	0x7301	SPF19	0x5301	ALTS	0x5303
PF20	0x7401	SPF20	0x5401	ALTT	0x5403
PF21	0x7501	SPF21	0x5501	ALTU	0x5503
PF22	0x7601	SPF22	0x5601	ALTV	0x5603

Table 6-2 Hexadecimal values for function and ALT keys *(Continued)*

PF	Hex	SPF	Hex	ALT	Hex
PF23	0x7701	SPF23	0x5701	ALTW	0x5703
PF24	0x7801	SPF24	0x5801	ALTX	0x5803
				ALTY	0x5903
				ALTZ	0x5a03

*Recommended entries.

Table 6-3 Hexadecimal values for application function keys

APP	Hex	APP	Hex
APP0	0x6002	APP32	0x4002
APP1	0x6102	APP33	0x4102
APP2	0x6202	APP34	0x4202
APP3	0x6302	APP35	0x4302
APP4	0x6402	APP36	0x4402
APP5	0x6502	APP37	0x4502
APP6	0x6602	APP38	0x4602
APP7	0x6702	APP39	0x4702
APP8	0x6802	APP40	0x4802
APP9	0x6902	APP41	0x4902
APP10	0x6a02	APP42	0x4a02
APP11	0x6b02	APP43	0x4b02
APP12	0x6c02	APP44	0x4c02
APP13	0x6d02	APP45	0x4d02

Table 6-3 Hexadecimal values for application function keys *(Continued)*

APP	Hex	APP	Hex
APP14	0x6de2	APP46	0x4e02
APP15	0x6f02	APP47	0x4f02
APP16	0x7002	APP48	0x5002
APP17	0x7102	APP49	0x5102
APP18	0x7202	APP50	0x5202
APP19	0x7302	APP51	0x5302
APP20	0x7402	APP52	0x5402
APP21	0x7502	APP53	0x5502
APP22	0x7602	APP54	0x5602
APP23	0x7702	APP55	0x5702
APP24	0x7802	APP56	0x5802
APP25	0x7902	APP57	0x5902
APP26	0x7a02	APP58	0x5a02
APP27	0x7b02	APP59	0x5b02
APP28	0x7c02	APP60	0x5c02
APP29	0x7d02	APP61	0x5d02
APP30	0x7e02	APP62	0x5e02
APP31	0x7f02	APP63	0x5f02

ASCII Character Mnemonics and Hex Values

Table 6-4 lists two- and three-letter ASCII mnemonics for control and extended control characters. It is derived from the include file `smascii.h`.

Table 6-4 ASCII character mnemonics and hexadecimal values

Mnemonic	Hex	Mnemonic	Hex	Mnemonic	Hex	Mnemonic	Hex
NUL	0x00	DLE	0x10			DCS	0x90
SOH	0x01	DC1	0x11			PU1	0x91
STX	0x02	DC2	0x12			PU2	0x92
ETX	0x03	DC3	0x13			STS	0x93
EOT	0x04	DC4	0x14	IND	0x84	CCH	0x94
ENQ	0x05	NAK	0x15	NEL	0x85	MW	0x95
ACK	0x06	SYN	0x16	SSA	0x86	SPA	0x96
BEL	0x07	ETB	0x17	ESA	0x87	EPA	0x97
BS	0x08	CAN	0x18	HTS	0x88		
HT	0x09	EM	0x19	HTJ	0x89		
LF	0x0a	SUB	0x1a	VTS	0x8a		
VT	0x0b	ESC	0x1b	PLD	0x8b	CSI	0x9b
FF	0x0c	FS	0x1c	PLU	0x8c	ST	0x9c
CR	0x0d	GS	0x1d	RI	0x8d	OCS	0x9d
SO	0x0e	RS	0x1e	SS2	0x8e	PM	0x9e
SI	0x0f	US	0x1f	SS3	0x8f	APC	0x9f
SP	0x20	DEL	0x7f				

Creating and Modifying a Key Translation File

To create or modify a key translation file:

1. Access the desired ASCII key translation file (or create a new one) using a text editor.
2. Edit the ASCII key translation file as required.
3. Use `key2bin` to convert the ASCII file to binary format.
4. If this is a new key translation file, include the path name of the binary file as the value of the `SMKEY` configuration variable. (The default value for `SMKEY` is set in the file pointed to by `SMVARS`. Refer to Chapter 2, "Application Variables," for information on changing configuration variables in setup files.)

Customizing Key Mapping

You can change the mappings of logical keys for the preferences of users or developers by:

- Changing the key translation file (which affects the entire application).
- Calling `sm_keyoption` to change the behavior at runtime. For more information, refer to "Key Change Function" on page 44-36 in *Application Development Guide*.

Example: Changing the Key Translation File

In the distributed key translation files for the PC, `XMIT` is mapped to the physical End key, and `NL` is mapped to the Enter key. The key translation file entries look like this:

```
XMIT(End) = NUL O

NL(Enter) = CR
```

`NUL O` is the character-sequence transmitted by the PC's End key; when

Panther receives this sequence, it carries out its `XMIT` function. Similarly, `CR` is transmitted by the PC's Enter key and Panther responds appropriately.

To use the Enter key for `XMIT` and the End key for NL, you can change the key translation file entries to read:

```
XMIT(Enter) = CR

NL(End) = NUL O
```

Using Alternate Key Translation Files

Many applications support more than one type of keyboard. With Panther you can provide this support without recompiling the application for each keyboard. Each terminal must have a working key translation file and video file (for character-mode only). List the path names for the key translation and video files in your application's `SMVARS` file. Assuming the `SMTERM` variable is set correctly, Panther selects the correct key translation file from the `SMVARS` file during initialization.

For example, the following excerpt is from the `SMVARS` file:

```
SMKEY = (hp|hp2392) $SMBASE/config/hpkeys.bin

SMKEY = (xterm) $SMBASE/config/xtermkeys.bin
```

CHAPTER

7 Video File

Panther is designed to run on many character-mode displays with numerous differences between them. For example, some displays are 80 columns wide, while others have 132 columns. Similarly, control sequences used to position the cursor and highlight data on the display often differ from model to model.

Panther uses a video file specific to a given terminal type to determine how to handle that terminal's display characteristics, and so display screens and messages correctly. Each video file defines the display characteristics for a given character-mode terminal type. The Panther distribution includes video files in the `config` directory that support around 35 character-mode terminal types. One of the distributed video files should work for you.

Note: GUI platforms do not require a video file.

This chapter covers the following topics:

- The Role of the Video File
- Video File Syntax—How to read video file entries and the concepts used to interpret them
- Creating and Modifying a Video File
- Video File Keywords—The keywords that can be included in a video file and commands that are needed to encode parameters
- Sample Video File

Finding the Correct Video File

As distributed, Panther can be configured to support around 35 character-mode terminals as well as those that emulate other terminals:

1. Identify the binary video file in the distributed `SMVARS` source file (in the `config` directory) that supports your terminal: among the `SMVIDEO` settings, find the one with a parenthetic mnemonic for your terminal type. For example, the `SMVIDEO` entry for a `vt100` terminal looks like this:

   ```
   SMVIDEO = (vt100) $SMBASE/config/vt100vid.bin
   ```

2. Set the `SMTERM` variable to the mnemonic—in the previous example, `SMTERM = vt100`. Or, enter the mnemonic when prompted for terminal type when you invoke `prodev`.

3. Invoke `prodev` and check whether basic functions work correctly.

   If the screen does not clear or characters are positioned incorrectly, try one of these approaches:

   - Use a different terminal mnemonic to access a different video file.
   - Create a video file from the `termcap` or `terminfo` databases (if available), or modify an existing video file. Convert the ASCII source file to binary with `vid2bin`.

   Test existing or modified video files until you find one that performs basic requirements correctly.

Table 7-1 provides a listing of popular terminal types. After you determine the family of terminals (there may be more than one video file to support a single family), try each of the files associated with the family of terminals by setting `SMTERM` to the appropriate terminal mnemonic.

Table 7-1 General categories of families of terminals

If the family is	Look for	Set SMTERM to
ANSI/DEC VT	`ESC` [or CSI in the ASCII video files	`vt100`

Table 7-1 General categories of families of terminals *(Continued)*

If the family is	Look for	Set SMTERM to
Televideo, Wyse	ESC * or ESC = in the ASCII video files	wy30; wy50
Hewlett-Packard	Video files supporting HP terminals begin with the letters hp.	hp
Data General	Video files supporting Data General terminals begin with the letters dg	dg214

The Role of the Video File

Panther video files are based on the termcap and terminfo databases. These databases were originally developed to handle display of the earliest full-screen editors such as emacs and vi. Because these editors use the screen non-sequentially, they need terminal-specific ways to move the cursor, clear the screen, insert lines, and so on. Although closely associated with UNIX, termcap and terminfo are also used on other operating systems, and list idiosyncrasies of many terminal types.

Text editors use visual attributes sparingly, if at all. Thus, termcap contains minimal information about handling them. Usually there are entries to start and end “stand-out” and sometimes entries to start and end “underline.” Notably missing are entries that explain how to combine attributes, like reverse video and blinking simultaneously. The terminfo database can combine attributes; in practice, however, the appropriate entries are usually missing.

Panther makes extensive use of attributes in all combinations, and supports color. Rather than extending termcap with additional codes, which might conflict with other extensions, Panther uses an independent file to describe the terminal specific information. Furthermore, some machines, notably PCs, lack terminfo capability.

Panther developed a set of commands that extend the limited set of commands used by `termcap` and abbreviates verbose sequences used by `terminfo`. Both syntaxes are supported. All the commands needed in the video file can be written using `terminfo` syntax; many can be written using the simpler `termcap` syntax and a few can benefit by using the extended commands.

A summary of the commands used to process parameters is described in this chapter; details and examples are also included.

The Basic Video File

At a minimum, a video file requires two entries:

- `CUP` for positioning the cursor.
- `ED` for erasing the display.

Although character-mode Panther functions with these two entries, it does so with a limited set of display capabilities—for example, visual attributes are unavailable. Speed can also be affected adversely—many video file entries serve to decrease the number of characters transmitted to the terminal.

Unless otherwise indicated, Panther assumes a 24-line by 80-column screen. Line 24 is used for status text and error messages, and the remaining 23 are available for screens. The non-display attribute is supported and available. The underline attribute is simulated by underscores placed wherever blanks appear in an underlined field. Clearing a line is done by writing spaces. Borders are available, and consist of printable characters only. Refer to page 7-18, “Enhancing a Basic Video File” for specific details on enhancing a basic video file to include display attributes, such as reverse, underline, and so on.

Processing Keywords—Automatic Parameter Sequencing

A stack is used to process a keyword in the video file—parameters are kept in a separate list. The stack is initially empty and parameters are generally pushed on the stack as needed. The parameters are ordered and a pointer is used to access them. It initially points to the first one. The stack is four levels deep; anything pushed off the

end is lost. There are commands that push a parameter or constant onto the stack, but no explicit pop commands. Output commands transmit the value on top of the stack, then remove it.

Arithmetic and logical operations take one or two operands from the top of the stack, and replace them with one result; thus, they perform an implicit pop. These types of operations use postfix notation. The operands are pushed, then the operation takes place. So the sequence `%p1 %p2 %p3 %+ %*` leaves *`parameter1`* + (*parameter2* * *parameter3*) on the stack. This same mechanism is used by `terminfo`.

Refer to page 7-8, "Parameters for Keyword Sequences" for information about specific parameters.

Supporting termcap commands

`termcap` commands do not use a stack mechanism. To support them, Panther uses an automatic parameter sequencing scheme where a current index into the parameter list is maintained. When a parameter is needed on the stack, the following actions occur:

- The current parameter is pushed and the index is incremented.
- If an output command is encountered and there is nothing on the stack to output, an automatic push is performed using the current index. The commands `%d %d` output two decimals; the sequence `%p1 %d %p2 %d` does the same thing.

The effect of this scheme is that `termcap`-style commands are translated into `terminfo`-style.

Although it is possible to use automatic sequencing and explicit parameter pushes in the same sequence, it is not recommended. Explicit pushes of constants with automatic parameter sequencing, however, is a useful combination. For example:

```
REPT= %p1 %c ESC F %'?' %p2 %+ %c
```

Video File Syntax

A video file is initially created and edited as an ASCII text file, which is accessible with any text editor. Lines that start with a pound sign (#) are treated as comments and ignored by the `vid2bin` utility when it converts the ASCII source file to binary format. All distributed video source files are documented with comments.

A video file consists of many instructions, one per line, and has the following format:

`keyword = variable-data`

keyword
: A single word used to define the instructions. Refer to page 7-19, "Video File Keywords."

  When you add entries to a video file, it is essential that you use the formats described for the specific video instruction. No error checking is done at runtime. The `vid2bin` utility checks for errors like missing, misspelled, and superfluous keywords, but ignores duplicate or conflicting entries.

variable-data
: A number, a list of characters, a sequence of characters, or a list of further instructions. The value of *`variable-data`* depends on the keyword. Refer to page 7-8, "Parameters for Keyword Sequences" for information on keywords that use specific parameters as variable-data.

Wrapping Lines

To continue a logical line on the next physical line, end the first line with a backslash. Do *not* leave a space between the backslash and carriage return. All white space (spaces and tabs) is skipped, except where noted. To enter a backslash as the last character of a line, use two backslashes (without spaces). Thus

text \	Continuation line
text \\	Ends with a backslash

text \\\	Backslash and a continuation

Double Quotes

A double quote `"` starts a string. Text between it and the next double quote (or the end of the line) is taken literally, including spaces. To include a double quote in a quoted string, use backslash quote `\"` with no space between.

“stty tabs”	Has an embedded space
stty tabs	No embedded space

Percent Sign (%)

The percent sign is a control character (refer to Table 7-3 for a list of percent commands). To enter a literal percent sign, enter it twice—for example, `%%`).

Inputting Control Characters

You can enter in a video file non-printing characters such as control characters with these strings:

- Any character as `0x` followed by two hexadecimal digits. For example, use `0x41` for `A`, or `0x01` for control-A. This method is useful for entering codes in the range `0x80` to `0xff` (extended ASCII control characters).
- A caret ^ followed by a letter or symbol to represent control characters in the range `0x01` to `0x1f`. Either `^A` or `^a` can represent `SOH (0x01)`. The symbols are `^[` for `ESC`; `^\` for `FS`; `^]` for `GS`; `^^` for `RS`; and `^_` for `US`.
- Two- and three-character ASCII mnemonics to represent control characters. The documentation provided with terminals often lists such sequences.

Refer to Table 6-4 for a list of mnemonics and hexadecimal values.

Extended ASCII control codes can be transmitted only if the communication line and terminal use eight data bits. Otherwise, the eight-bit code can be replaced by two seven-bit codes—the first code is `ESC` (`0x1b`), the second is `0x40` *less* than the desired

eight-bit control character. For example, CSI (0x9b) is replaced by ESC 0x5b, or ESC [. If your video file contains extended ASCII control codes, Panther assumes they can be used; it does not transmit the two-character sequence automatically.

Note: Some computers internally toggle the high bit of a character. The numbers given in this guide are always standard ASCII.

Parameters for Keyword Sequences

Certain keywords take values or sequences that cannot be completely specified in advance. For example, the cursor position sequence requires the line and column number before moving. The commands using these sequences are passed extra parameters.

Table 7-2 lists those keywords that are passed parameters. The number in parentheses is the number of parameters for each keyword.

Table 7-2 Keywords and expected parameters

Keyword	Action	Parameters
REPT	repeat sequence (2)	Character Number of times to repeat
EW	erase window (5)	Start line Start column Number of lines Number of columns Background color
CUP	cursor position (2)	Line and column (relative to 0)
CUU	cursor up (1)	Line increment
CUD	cursor down (1)	Line increment
CUF	cursor forward (1)	Column increment
CUB	cursor backward (1)	Column increment
SGR	set latch graphics rendition (12)	See page 7-38, SGR
ASGR	set area graphics rendition (12)	See page 7-31, ASGR

Percent Commands

Parameters are encoded in sequences by percent commands where the sequence starts with the % symbol. Percent commands perform these tasks:

- Cause data to be output.
- Are used for control purposes.

Panther uses a stack mechanism like that used by terminfo; refer to page 7-8, "Parameters for Keyword Sequences" for a description. However, use percent commands with care, because all sequences go through the same processing, even those that do not use runtime arguments. In particular, to enter a percent sign as a literal, you must use %%.

Percent commands are summarized in Table 7-3. They are organized by function, and their source is indicated (C for termcap; I for terminfo; E for Panther extended command). Descriptions and examples are provided in subsequent sections.

Percent commands that take a count (represented by a # immediately after a %) do not need to have a count specified. If none is specified, the count is assumed to be 1. For example, %w is equivalent to %1w. (This does not apply to %d.)

Table 7-3 Percent commands

Percent command	Source*	Description
Output Commands (page 7-11)		
%%	C / I	Output a percent sign
%.	C	Output a character
%c	I	Output a character
%d	C / I	Output a decimal
%#d	I	Output a #-digit decimal, blank filled
%0#d	I	Output a #-digit decimal, zero filled, like the termcap %2 which is not supported
%+	C	Add and output a character
%#z	E	Output # (decimal number) binary zeros

Table 7-3 Percent commands *(Continued)*

Percent command	Source*	Description
%#w	E	Wait (sleep) # seconds
%S	E	Issue a system command
Stack Manipulation and Arithmetic Commands **(page 7-12)**		
%p#	I	Push parameter # (1 - 12 allowed)
%'c'	I	Push the character constant c
%{#}	I	Push the integer constant #
%+	I	Add
%-	I	Subtract
%*	I	Multiple
%/	I	Divide
%m	I	Modulus
%|	I	Bitwise OR
%^	I	Bitwise exclusive OR
%&	I	Bitwise AND
%=	I	Logical EQUAL TO
%>	I	Logical GREATER THAN
%<	I	Logical LESS THAN
%!	I	Logical NOT
%~	I	One's complement
Parameter Sequencing Commands **(page 7-13)**		
%#u	E	Discard # parameters
%#b	E	Back up # parameters
%i	C / I	Increment the next two parameters

Table 7-3 Percent commands *(Continued)*

Percent command	Source*	Description
%r	C	Reverse the next two parameters
Control Flow Commands **(page 7-14)**		
%? *expr* %t *then-part* %e *else-part* %;	I	Conditionally execute one of two command sequences
expr %t *then-part* %e *else-part* %;	E	Same effect as previous
%#(... %)	E	Repeat the sequence # times
%l(... %)	E	Select operations from a list
terminfo commands not supported		
%P, %g		Letter variables
$<#>		Padding (use %#z instead) (page 7-16)
* C = termcap; I = terminfo; E = Panther extended command		

Output Commands

%%	Outputs a literal percent sign.
%.	Outputs a character, like `printf`; this command is supplied for `termcap` compatibility.
%c	Outputs a character, like `printf` (equivalent to %).
%d	Outputs a decimal, any number of digits, no fill. It has variations that allow for specifying the number of digits, and whether blank-fill or zero-fill is to be used.
%#d	Outputs a #-digit decimal, blank filled. For example, `%3d` outputs at most three decimal digits with blank fill.

%0#d	Outputs a #-digit decimal, zero filled. For example, `%03d` outputs at most three decimal digits with zero fill.
%+	Adds and outputs a character. If the stack is empty, the character following %+ is added to the next parameter, the sum is output as a character, and the parameter index is incremented. Also refer to definitions of %+ in arithmetic commands and automatic parameter sequencing.
%#z	Outputs the specified number of NUL characters (binary zero). It is usually used for padding, to insert a time delay for commands such as erase screen. The sequence `%100z` outputs 100 pad bytes to the terminal.
%#w	Waits (sleeps) the specified # of seconds. (Although supported on all UNIX platforms, it is not supported on systems where the sleep library routine is unavailable). It is often used as a time delay for `INIT` and `RESET` sequences. The sequence `%2w` evokes a wait of two seconds.

%S	Issues a system command. The format is `%S (`*string* `%)`—the string is passed to the command interpreter for execution. To include spaces in the string, enclose the text in single quotation marks. The following examples illustrate two ways of making a system call `stty tabs`:
	`%S('stty tabs'%)`
	`%S(stty SP tabs%)`

Stack Manipulation and Arithmetic Commands

Commands are available to push parameters and constants. Only four levels of stack are supported, and anything pushed off the end is discarded.

%p# .	Pushes parameter #; 1 to 11 is allowed. For example, the sequence `%p2` pushes the second parameter
%'*c*'	Pushes the character constant *c*. For example, the sequence `%'x'` pushes the character x.
%{#}	Pushes the integer constant #. For example, the sequence `%{12}` pushes the number 12.

Various arithmetic and logical operations such as `%+, %/, %&, %>` are supported. They require one or two operands on the stack. If necessary an automatic push is generated, using the next parameter.

`%'@' %| %| %| %c` Bitwise OR 3 parameters with `@`, then output result

The automatic parameter sequencing mechanism works well in the above example. Because bitwise OR requires two parameters and there is only one on the stack, a push is performed. No push is required to process `%c` because an entry already exists on the stack. Thus only three parameters are consumed and the result of the bitwise OR is output.

In the following sequence the first parameter is pushed, then a space character (`0x20`) is pushed. The `%+` command pops and adds these values and puts the answer on the stack. `%c` then pops this value and transmits it to the terminal.

```
%p1 %'SP' %+ %c
```

Parameter Sequencing Commands

With automatic sequencing of parameters, sometimes it is necessary to access the parameters in a different order. The following percent commands can be specified:

`%#b`	Backs up the # of parameters by decrementing the parameter index. For example, the following sequences output the same parameter twice:
	`%d %b %d`
	`%p1 %d %p1 %d`

`%#u`	Uses up or discards # of parameters by incrementing the parameter index. For example, the following sequences output in reverse order:
	`%u %d %2b %d`
	`%p2 %d %p1 %d`

Parameter Changing Commands

Parameter changing commands either increment parameters or reverse them

`%i`	Increments the next two parameters; however, no output is performed and no parameters are consumed. It is used almost exclusively in `termcap` cursor positioning sequences. It is passed line and column parameters, with the upper left being (0,0). Many terminals expect the line and column to be relative to (1,1). The following sequence adds one to each parameter and sends it out as decimals: ESC [%i %d ; %d H

%r	Reverses the next two parameters. It is unnecessary if explicit parameter pushes are used; in fact, it should be avoided in that case because the numbering of the parameters is reversed. This command is often used in cursor positioning sequences where the terminal requires that column be given first and then the line (the default being the other way around). For example, this sequence outputs column first, and then line: FS G %r %c %c

Control Flow Commands

Control flow commands conditionally execute command sequences. Some specify the number of times to repeat the sequence, and others select operations from a list.

Conditional Command

%? expr %t then-part%e else-part%;

This is the general if-then-else clause, which can be abbreviated by omitting the *if*:

expr %t then-part %e else-part %;.

expr is any sequence including the empty sequence. The %t, which is required, pops a value from the stack and tests it, executing *then-part* if it is true (non-zero) and *else-part* otherwise.

then-part and *else-part* can be any sequence, including the empty sequence. If *else-part* is empty, %e can be omitted. They can also contain conditionals, so else-if can be implemented. However, this can produce undecipherable sequences. It is provided for terminfo compatibility. The list command (below) is an alternative.

If %t finds the stack is empty, it generates an automatic push of the next parameter. %t consumes one parameter; however, the incrementing of the parameter index is delayed until after the entire conditional is executed. A conditional always consumes exactly one parameter, regardless of which branch is taken or of the content of *then-part* or *else-part*. If either of those use automatic parameter sequencing, they use a local index. Thus even if they consume two parameters, at the end of the conditional the parameter index is reset. When the next command is reached, only one parameter has been consumed.

In this sequence, %t ; %c %;, one parameter is consumed. It outputs and a character if the parameter is non-zero, otherwise it skips the parameter.

In the next example, the constant (binary) 1 is pushed, the parameter is compared with 1, and the boolean value is left on the stack. If the value is true, nothing is done; otherwise the parameter is output as a decimal.

```
%? %{1} %p1 %= %t %e %p1 %d %;
```

The following sequence exhibits a simple "either-or" condition that is sometimes used to toggle an attribute on or off. It outputs `ESC` (if the parameter is non-zero, and `ESC`) otherwise:

```
ESC %t ( %e ) %;
```

Repeat Command

```
%# (... %)
```

Performs the same action for several parameters. It is used with automatic parameter sequencing, and is almost useless if explicit parameter pushes are used. The count is specified after the percent sign. All the commands between %# (and %) are executed the specified number of times. For example, the first outputs three decimals, and the second outputs up to three non-zero parameters:

```
%3( %d %)
%3( %t %d %; %)
```

List Command

```
%l(... %)
```

Available as an alternative to an if-then-else-if construction, but is seldom needed. It implements a simple select or case conditional. The general format is:

```
%l(value1: expr1 %; value2: expr2 %; ... %)
```

The values are single character constants representing the various cases. The expression is executed if the value matches the top of stack. At most one expression is executed (i.e., each case contains a break). If the value is missing, the expression is evaluated as a default. For correct operation, the default case must occur last in the list. The colons do not have a leading percent sign, so no selector can be a colon. The %; after the last element of the list is not required.

The parameter on the stack (automatically pushed, if necessary) is popped and tested against the various cases. The parameter index is incremented by one after the entire list is processed, even if the expressions use parameters. The following sequence outputs nothing if the parameter is 0; `ESC` if it is 1; `FS` otherwise: `%l( 0:%; 1:ESC%; :FS %)`

Padding

Certain terminals (or tty drivers) require extra time to execute instructions. Some terminal documentation specifies the delay required for each command. If random characters appear on the screen, particularly characters that are part of command sequences, time delays may be required. Delays can be introduced in two ways:

`%#w`	Causes a wait (sleep) for the specified number of seconds. This command is usually only required in terminal initialization or reset sequences. A hard reset of a terminal sometimes requires a sleep of one or two seconds. The following sequence causes a two second sleep after terminal reset: `ESC c %2w`
`%#z`	Outputs the specified number of zeros. This command is occasionally needed on the erase display or erase line commands and very rarely in the cursor positioning sequence. The number of zeros to send may range from about five, for very short delays, to several thousand for longer delays. Usually 100 or so is adequate. The following sequence specifies 100 pad zeros after clearing the screen: `ESC [ J %100z` `termcap` indicates padding by using a number at the beginning of a sequence, which is the number of milliseconds of pad required. `terminfo` uses the syntax `$<#>`. It is easy to convert to the `%#z` notation, because at 9600 baud, one character takes one millisecond to output.

Creating and Modifying a Video File

If none of the distributed video files is suitable for your terminal type, you can create one with one of these methods:

- Modify one of the video source files in the Panther distribution. Review your terminal's documentation to determine whether it is similar to any of the supported terminal types.
- Use the `term2vid` utility to create a video file from your terminal's `termcap` or `terminfo` data.

After you have a video file that displays Panther correctly on your terminal, you can enhance it to provide greater functionality.

Modifying an Existing Video File

It is easiest to create a video file by basing it on an existing one that you modify:

1. Identify the desired video file by setting the `SMTERM` variable in your environment to the mnemonic (for example, `SMTERM = vt100`) for the selected video file. (Or leave it unset and type the name in when `prodev` prompts you for it.)
2. Execute `prodev` and retrieve an existing screen or try creating one.
3. Check a few basic functions: Does the screen clear? Are characters positioned properly (or even close)?

   If things seem to work closely to the way you would expect them to—you probably have a good, modifiable video file.
4. If the video file works well, but the keys are not translating properly, modify the key translation file or create a new one that is compatible with the video file (refer to Chapter 6).
5. Make a copy of the best working video file you found; name it `testvid`.
6. Edit the ASCII video file using any text editor. Comment out or delete all lines except for the `ED` (erase display) entry and `CUP` (cursor position) entry.
7. Convert the ASCII file to binary format using `vid2bin`.
8. Edit the `SMVARS` source file. Add the following entry after the last `SMVIDEO` entry:

   ```
   SMVIDEO = testvid.bin
   ```
9. Convert the `SMVARS` source file to binary with `var2bin`.

10. Invoke `prodev` and check the basic functions.

Creating a Video File

The `term2vid` utility uses your system's `termcap/terminfo` entry to create a rudimentary video source file. The utility has this syntax:

`term2vid [-f]` *`terminalType`*

`-f` allows the output to overwrite an existing file; *`terminalType`* is the name of the terminal type, the value of the system `TERM` variable.

The ASCII output file is named after *`terminalType`*. If necessary, modify the source to include features not listed in the source `termcap/terminfo`. After you edit the source, convert it to binary format with `vid2bin` and test it.

Enhancing a Basic Video File

After you have a basic working video file, you can enhance it to provide greater functionality. Incrementally add and test various features until you have a fully operational video file. The easiest way to add the recommended sequences is to copy them from an existing video file or, if available, from your terminal's documentation. The following features are commonly added:

- Cursor movement control: add an entry for `CMFLGS`. (See page 7-25, "`CMFLGS`.")
- Display attributes (colors, reverse, underline) with `SGR`, `ASGR`, `LATCHATT`, `AREAATT` entries. (See page 7-28, "Display Attributes.")
- A way to turn the cursor on and off so that menus display properly: `CON` and `COF` entries. (See page 7-27, "`COF`.")
- Entries that enable use of graphic characters to draw lines and borders. (See page 7-41, "Graphics and International Character Support.")

  `MODE0...MODE6`

  `GRAPH`

  `BORDER`

  `BOX`

ARROWS

Video File Keywords

Table 7-4 includes a list of all video file entry keywords, arranged by function. Detailed information on each keyword is available on the indicated pages. Refer to the sample video file for syntax examples.

Table 7-4 Video file keywords

Keyword	Description
Basic Capabilities (page 7-22)	
BOTTOMRT	Last position of screen may be written without scrolling the display
BUFSIZ	Number of characters to accumulate before flushing
COLMS	Number of columns on screen
INIT	Initialization sequence
KBD_DELAY	Timing interval for keyboard input
LINES	Number of lines on screen
REPMAX	Maximum number of repeated characters
REPT	Repeat the following character sequence
RESET	Undo initialization sequence
Screen and Line Erasure (page 7-25)	
ED	Erase entire display
EL	Erase to end of current line
EW	Erase window

Table 7-4 Video file keywords *(Continued)*

Keyword	Description
Cursor Position (page 7-25)	
CMFLGS	Allowed cursor-motion shortcuts
CUB	Cursor backward
CUD	Cursor down
CUF	Cursor forward
CUP	Absolute cursor position
CUU	Cursor up
Cursor Appearance (page 7-27)	
COF	Turn cursor off
CON	Turn cursor on
INSOFF	Overstrike-mode cursor
INSON	Insert-mode cursor
RCP	Restore cursor position and attribute
SCP	Save cursor position and attribute
Display Attributes (page 7-28)	
AREAATT	List of available area attributes
ARGR	Remove area attribute
ASGR	Set graphics rendition (area)
COLOR	List of colors
EMPHASIS_KEEPATT	Specify attributes retained for grayed objects
EMPHASIS_SETATT	Set attributes for grayed objects
LATCHATT	List of available latch attributes
SGR	Set graphics rendition (latch)

Table 7-4 Video file keywords *(Continued)*

Keyword	Description
SPXATT	List of attributes that do not affect space
Status Line (page 7-39)	
CMSG	Close status line
MSGATT	Status line attributes
OMSG	Open status line
Graphics and International Character Support (page 7-41)	
GRAPH	Graphics character equivalents
GRTYPE	Shortcut for defining graphics characters
MODE0	Normal character set sequence
MODE1	Locking shift to alternate character set 1
MODE2	Locking shift to alternate character set 2
MODE3	Locking shift to alternate character set 3
MODE4	Non-locking shift to alternate character set 1
MODE5	Non-locking shift to alternate character set 2
MODE6	Non-locking shift to alternate character set 3
Borders and Line Drawing (page 7-44)	
BORDER	Characters that make up the 10 border styles
BOX	Characters that make up the styles for box and line drawing
BRDATT	Available border attributes
Indicators (page 7-46)	
ARROWS	Indicator characters for shifting and scrolling
BELL	"Visible bell" alarm sequence
CBDSEL	Deselection character for groups

Table 7-4 Video file keywords *(Continued)*

Keyword	Description
CBSEL	Selection character for groups
MARKCHAR	Character used for check menu items
OPTMNUIND	Character used as an option menu indicator
SUBMNSTRING	String on menu item indicating presence of submenu
Drivers **(page 7-48)**	
BLKDRVR	Name of block mode driver
MOUSEDRIVER	Name of mouse driver
Miscellaneous **(page 7-48)**	
COMPRESS	Output data compression for J*term*
CURPOS	Display the current cursor position on the status line

Basic Capabilities

BOTTOMRT

A flag indicating that the bottom right-hand corner of the display can be written to without causing the display to scroll. If this flag is not present, Panther does not write to that position.

BUFSIZ

Sets the size of the output buffer used by Panther. If it is omitted, Panther calculates a reasonable default size. Include this entry only if special circumstances warrant. For example, if you make extensive use of a screen-oriented debugger, you can set BUFSIZ to a large value; that effectively disables the delayed-write feature, which can interfere with debugging.

COLMS

Indicates the number of columns on the display. The default value is 80. In some windowing environments (for example, Sun workstations) the values of LINES and COLMS are overridden by the number of lines and columns in the active window.

`INIT`

Terminal initialization sequence, output by the library function `sm_initcrt`. There is no default; and the keyword can be omitted. `INIT` is typically used to change the mode of the terminal, to map function keys, select attribute styles, and so on. Padding is sometimes required, either with `%#z or %#w`.

```
# map 2 function keys, then wait 2 seconds

INIT = %S( "/etc/keyset f1 ^a P ^m" %) \
       %S( "/etc/keyset f2 ^a Q ^m" %) \
       %2w

# load alternate character sets

INIT = ESC)F ESC*| ESC+}
```

Specifying Cursor Style

On ASCII terminals, you can use escape sequences for specifying the cursor style in the `INIT` and `RESET` strings in the normal fashion. The format is

```
INIT = C n1, n2[, n3, flag]

RESET = C n1, n2[, n3]
```

n1 and *n2* specify the top and bottom scan lines respectively for the cursor block; with line 0 at the top. (A *scan line* is the smallest vertical unit on your display—one pixel wide). The optional n3 specifies the blink rate, as follows:

1	no cursor
2	fast blink (the default)
3	slow blink
0	fast blink (Not always valid on non-IBM systems)

The standard sequences for a blinking block cursor are

For a monochrome monitor:	INIT = C 0, 13, 0
For a CGA monitor:	INIT = C 0, 7, 0
For a EGA monitor:	INIT = C 0, 13, 2

`KBD_DELAY`

Assigns a timing interval to keyboard input. A positive integer between 1 and 10 represents an interval in tenths of a second. A negative integer or 0 represents an interval of indefinitely great length.

If you use key sequences that are lead-ins of other sequences, you must assign a timing interval via `KBD_DELAY` to determine when a key sequence ends.

`LINES`

Indicates the number of lines on the display. The default value is 24. In general one line is reserved for status and error messages so the maximum screen size is usually one less than the number specified here. (Refer to `OMSG and CMSG` on page 7-39 for exceptions.)

`REPMAX`

Indicates the maximum number of characters `REPT` can repeat. To determine the proper value of `REPMAX`, omit the entry from the video file; then using `prodev`, create a field that extends the entire width of the screen. Once you choose `XMIT`, if the entire field has the underline attribute, you do not need the `REPMAX` entry. If the field is not underlined, gradually shorten the field until the underlines fill the field. The resulting number determines the largest possible value of `REPMAX`.

`REPT`

A repeat-character sequence that is passed two parameters: the character to be repeated and the number of times to display it. There is no default, because most terminals do not support character repeat. If it is available on your terminal, include the `REPT` entry. The repeat sequence is used whenever possible, usually for borders and for clearing areas of the screen. If borders do not appear correctly, your sequence could be incorrect. A repeat sequence is not used to repeat a control character, and it never extends to more than one line.

For example, the following entry outputs the character, then `ESC F` and the count with 0x3f (the ASCII value of '?') added:

```
REPT= %c ESC F %+ ?
```

The next entry sets the maximum count for the preceding `REPT` entry. Anything over this count splits into more sequences.

```
REPMAX= 64
```

`RESET`

A reset-terminal sequence, output by the library function `sm_resetcrt`. There is no default. The `RESET` sequence should be set to undo the effects of

`INIT`. For many terminals a hard reset that resets the terminal to the state stored in non-volatile memory is appropriate.

Screen and Line Erasure

`ED`

Provides the control sequence that erases the display. This entry is required and clears all available display attributes, including background color. You can find the correct command in your terminal manual or in `termcap` as `cl`. Some terminals require padding after this command. The following example is common for ANSI terminals

```
ED = ESC [ J
```

`EL`

Provides a sequence that erases characters and attributes from the cursor to the end of the line. If `EL` is not in the video file, Panther erases the line by writing blanks. You can find the sequence in `termcap` as `ce`. Some terminals require padding after this command. The first example is common for ANSI terminals; the second illustrates a padded entry:

```
EL = ESC [ K
EL = ESC [ 0 K %100z
```

`EW`

Provides a sequence that erases a rectangular region on the screen, to a given background color if available. Five parameters are passed: start line, start column, number of lines, number of columns, and background color. (If color is not available, the fifth parameter is ignored.)

Cursor Position

`CMFLGS`

Lists shortcuts Panther uses for cursor positioning:

`CR`	Carriage return (`0x0d`, or `^M`) moves the cursor to the first column of the current line.
`LF`	Linefeed (`0x0a`, or `^J`) moves the cursor down one line in the same column.

BS	Backspace (0x08, or ^H) moves the cursor one position to the left with out erasing anything.
AM	Automatic margin: the cursor automatically wraps to the first column when it reaches the right-hand edge of the display.

Note: The AM setting of CMFLGS must match the auto-wrap setting of the terminal. If the setting is not correct, the terminal display may be irregular. Consider using INIT and RESET to turn terminal auto-wrap on and off as desired.

Most terminals are capable of the first three. The fourth can frequently be found in termcap as am. It cannot be used on terminals with the xen1 glitch (i.e., VT100-style delayed auto margin).

CUB

Moves the cursor backward in the same row. Takes the number of columns to move as a parameter. Do not specify this keyword if the sequence can only move the cursor one column at a time. The following entry moves the cursor back:

```
CUB = ESC [ %d D
```

CUD

Moves the cursor down in the same column. Takes the number of lines to move as a parameter. Do not specify this keyword if the sequence can only move the cursor one line at a time. The following entry moves the cursor down:

```
CUD = ESC [ %d B
```

CUF

Moves the cursor forward in the same row. Takes the number of columns to move as a parameter. Do not specify this keyword if the sequence can only move the cursor one column at a time. The following entry moves the cursor forward:

```
CUF = ESC [ % C
```

CUU

Moves the cursor up in the same column. Takes the number of lines to move as a parameter. Do not specify this keyword if the sequence can only move the cursor one line at a time. The following entry moves the cursor up:

```
CUU = ESC [ %d A
```

`CUP`

Establishes absolute cursor position which is required to run character-mode Panther. This sequence appears in `termcap` as `cm`. It takes two parameters: the target line and the target column, in that order and relative to 0. `%i` (increment) is used to convert them to be relative to one. ANSI terminals need the line and column as decimals. Other terminals add a fixed value to the line and column to make them printable characters; `%+` is used. The following example is for an ANSI terminal:

```
CUP = ESC [ %i %d;%d H
```

Another common scheme is to output the line and column as characters, after adding `SP`. The entry might look like the following example:

```
CUP = FS C %+SP %+SP
```

Cursor Appearance

`COF`

Turns the cursor off. If possible, both `COF` and `CON` should be specified. Menus (using a block cursor) look better with the regular cursor off. Also Panther often must move the cursor around the screen to put text in fields, to scroll arrays, and so on. If the cursor is off during these operations, the user is not disturbed by its flickering all over the screen.

`CON`

Turns the cursor on in the desired style. A blinking block cursor is recommended because an underline cursor is difficult to see in an underlined field.

Note: You can use the `INIT` and `RESET` sequences to switch between the cursor style used in Panther applications and that used on the command line.

Many terminals have no ability to turn the cursor on and off. Although Panther attempts to minimize cursor movement, some flickering is unavoidable.

`CON` and `COF` can sometimes be found in the terminal manual as cursor attributes and in `termcap` as `CO` and `CF`. The following entries are an example of a pair of `COF` and `CON` sequences for some ANSI terminals:

```
CON = ESC [>5l
COF = ESC [>5h
```

`INSOFF` and `INSON`

`INSOFF` and `INSON`, change the cursor style so that you can easily see which mode you are in: insert or overstrike. By convention, the insert cursor is about one-half the size of the regular, overstrike cursor. `INSOFF` or `INSON` is issued to the terminal when you toggle Panther's data entry mode using the `INSERT` key. On many terminals, changing the cursor style also turns it on; in this case, the `INSOFF` is the same as `COF`, so you can omit it altogether. If the cursor style can be changed without turning it on or off, use both `INSON` and `INSOFF`. Uses the same escape sequence format as `INIT` and `RESET`.

`RCP` and `SCP`

Saves (`SCP`) and restores (`RCP`) cursor position and attribute.

Display Attributes

Panther supports highlight, blink, underline and reverse video attributes. If either highlight or blink is not available, low intensity is supported in its place. An additional attribute, standout, can be assigned to any other desired attribute—for example dim or italics, if available. The display attribute keywords `AREAATT` and `LATCHATT` are used to list the attributes available in each style and associate a character with each attribute.

Attribute Types

Panther supports three different kinds of attribute handling.

- latch attributes—Assigns attributes to any characters written after the current cursor position. Latch attributes require no space on the screen. ANSI terminals use this method.
- area attributes—Assigns attributes to all characters from the cursor position to the next attribute (or end of line or end of screen). Area attributes do not occupy a screen position (they are “non-embedded” or “no space”). In this style, Panther positions the cursor to the end of the area to be changed, sets the ending attribute, then positions the cursor to the beginning of the area and sets its attribute.
- onscreen attributes—Act like area attributes, but occupy a screen position. (They are “embedded” or “spacing.”) This style of attribute handling dictates that fields and/or display areas cannot be adjacent, because a space must be

reserved for the attribute. Display of windows may be hampered by lack of space for onscreen attributes.

You can set several modes on your terminal. Many terminals support both area and onscreen attributes. If so, you should select area (“non-embedded” or “no space”) rather than onscreen (“embedded” or “spacing”) attributes. Some terminals support one latch attribute and several area attributes simultaneously.

If your terminal has only one attribute style available, it is recommended that you select reverse video. Panther supports non-display in software, so you can omit that attribute. Underlines are simulated (by writing an underscore character) if that attribute is not available.

You can find attribute information in your terminal's documentation or, perhaps, in your `termcap` database (if applicable). The codes `so`, `ul` and `bl` specify standout (usually reverse video), underline, and bold respectively. The codes se, ue and be specify the sequence to end the attributes. The standard ANSI sequences are:

```
so=\E[7m:se=\E[0m:ul=\E[4m:ue=\E[0m:bl=\E[1m:be=\E[0m
```

If you find something like these, ANSI latch attributes are available. If you find entries `ug#1:sg#1`, onscreen attributes are available.

`AREAATT`

Lists the area or onscreen attributes that are available, and associates a character with each. The possible attributes are:

`ACS`	Alternate character set (line drawing graphics)
`BLINK`	Blink or other standout
`DIM`	Dim (low intensity)
`HILIGHT`	Highlight (bold)
`REVERSE`	Reverse (or inverse) video
`STANDOUT`	User selected standout mode
`UNDERLN`	Underline

In addition, flags are available that specify how the attributes are implemented by the terminal. The flags are:

LINEWRAP	The attribute wraps from line to line
ONSCREEN	The attribute uses a screen position
REWRITE	Must rewrite attribute when writing character
SCREENWRAP	The attribute wraps from bottom of screen to top

Area and onscreen attributes modify all characters from the start attribute to the next attribute or to an `end', whichever is closer. (An `end' is either the end of the screen or the end of the line.) If there is no `end', use SCREENWRAP. If the end is the end of screen, use LINEWRAP. If end is the end of the line, omit both wrap flags. Some terminals allow you to select the style. For onscreen attributes, SCREENWRAP is best and LINEWRAP a good second; for area attributes the choices are about the same. If the attribute takes up a screen position, use the ONSCREEN flag.

```
AREAATT= BLINK= 2 DIM= p REVERSE= 4 UNDERLN= 8 \
         ONSCREEN LINEWRAP

ASGR = ESC G %u %'0' %5( %| %) %c
```

On some terminals writing a character at the position where an attribute was set can remove the attribute. Immediately after placing the attribute, the character may be written with no problems; however, the next time a character is written there, the attribute disappears. In this case, use the REWRITE flag to reset the attribute before writing to that position. The following example illustrates how the REWRITE flag is used (in Televideo 925 video file):

```
AREAATT = REVERSE = 4 UNDERLN = 8 BLINK = 2 REWRITE

ASGR = ESC G %'0' %9( %| %) %c
```

Some terminals restrict the number of attributes, use an ARGR entry to remove attributes.

ARGR

Removes area attributes. Some terminals restrict the number of attributes (as set with ASGR) that are available on a given line. If possible, use an ARGR entry. Changing an attribute to normal does not remove it; a normal attribute stops the propagation of a previous attribute only. The following example illustrates how the ARGR entry might appear in your video file:

```
AREAATT = REVERSE = Q UNDERLN = `
ASGR = ESC d %u %'@' %5( %| %) %c
```

ARGR = ESC e

Panther uses the ARGR entry to remove repeated attributes, to avoid exceeding the maximum number of attributes on a line. If there is no maximum, the remove attribute sequence can be omitted; it sometimes makes the screen "wiggle."

If the maximum number of attributes is small, Panther's performance can be limited. ARGR is desirable because having many attributes onscreen can dramatically slow performance, because Panther must keep rewriting them as attributes change.

ASGR

An area set graphics rendition sequence, used in conjunction with AREAATT, is passed twelve parameters. The first nine are the same as used by terminfo. The parameters, in order, represent:

1	standout
2	underline
3	reverse video
4	blink
5	dim (low intensity)
6	highlight (bold)
7	blank
8	protect not used, always 0
9	alternate character
10	foreground color (if available)
11	background color (if available)
12	background highlight

If an attribute is desired, the parameter passed is the character associated with the attribute, as explained in the description of SGR. If the attribute is not desired, the parameter passed is (binary) 0.

If no attributes are specified in the video file, Panther supports only two attributes: non-display (done in software anyway) and underline (using the underscore character).

`COLOR`

Used to associate a character with each color, just as `LATCHATT` associates a character with each attribute. Panther supports eight foreground and background colors. The `CTYPE` entry has flags that tell Panther what background color is available. You only need to specify the three primary colors in the video file. If other colors are not specified, they are generated according to the following rules:

BLACK =	RED & GREEN & BLUE
BLUE	Must be specified
GREEN	Must be specified
CYAN =	GREEN | BLUE
RED	Must be specified
MAGENTA =	RED | BLUE
YELLOW =	RED | GREEN
WHITE =	RED | GREEN | BLUE

The tenth parameter to `SGR` or `ASGR` is the character representing the foreground color; the eleventh represents the background color (it is 0 if background color is not available). Many ANSI terminals set foreground color with the following sequence: `ESC [ 3x m –` where `x` ranges from 0 for black to 7 for white. Background color is often set with `ESC [4x m`. The order of the colors varies from terminal to terminal.

On color terminals, `REVERSE` often means black on white. If background color is available, Panther prefers that `REVERSE` is not specified in the video file. It uses the specified color as the background, and either black or white as the foreground. The following example is suitable for a color ANSI terminal:

```
LATCHATT = HILIGHT = 1 BLINK = 5
COLOR = RED = 4 GREEN = 2 BLUE = 1 BACKGRND
SGR = %3u ESC [ 0 %3( %?%t ; %c %; %) ; %3u 3%c ; 4%c m
```

If the terminal has a unique sequence for each color, a list command works well. In the following example, the ANSI attribute sequence `ESC [ 0 ; p1 ; p2 ; ... m`) is used:

```
LATCHATT = REVERSE = 7 HILIGHT = 2
COLOR = CYAN = 0 MAGENTA = 1 BLUE = 2 YELLOW =3 \
        GREEN = 4 RED = 5 BLACK = 6 WHITE = 7
SGR = ESC [ 0 %p3%t;7%; %p6%t;2%; \
      %l( 0:;>1%; 1:;5%; 2:;5;>1%; 3:;4%; \
      4:;4;>1%; 5:;4;5%; 6:;4;5;>1 %) m
```

The values for the colors are:

```
cyan     >1
magenta  5
blue     5 ; > 1
yellow   4
green    4 ; > 1
red      4 ; 5
black    4 ; 5 ; > 1
```

Some terminals use `ESC [ 2 ; x ; y m` to set color and other attributes—*x* is the foreground color and y is the background color; both numbers range from 0 to 7. If highlight is desired in the foreground, `8` should be added to *x*. If blink is desired, 8 should be added to y. The following video entries satisfy these requirements:

```
LATCHATT = HILIGHT = 8 BLINK = 8
COLOR = RED = 4 GREEN = 2 BLUE = 1 BACKGRND
SGR = ESC [ 2 ; %p10 %p6 %+ %d ; %p11 %p4 %+ %d m
```

`EMPHASIS_KEEPATT`

Use this entry to specify the attributes to be retained when implementing drop shadows and grayed objects. By default, all attributes are enabled except `HILIGHT`. This variable is used in conjunction with the `EMPHASIS` setup variable. You can change this setting at runtime with the library function `sm_pset`. Refer to Table 2-1 on page 2-4 for a list of display attribute keywords.

`EMPHASIS_SETATT`

Use this entry to specify the attributes to apply when implementing drop shadows and grayed objects. By default, this variable has two attributes enabled: `REVERSE` and `DIM`. This variable is used in conjunction with the `EMPHASIS` setup variable. You can change attributes at runtime with the library function `sm_pset`. Refer to Table 2-1 on page 2-4 for a list of display attribute keywords.

`LATCHATT`

Lists the available attributes, and associates a character with each. The possible attributes are:

`ACS`	Alternate character set (line drawing graphics)
`B_HIGLIGHT`	Background highlight
`BLANK`	Non-display (foreground not shown)
`BLINK`	Blink or other standout
`DIM`	Dim (low intensity)
`HILIGHT`	Highlight (bold)
`REVERSE`	Reverse (or inverse) video
`STANDOUT`	User selected standout mode
`UNDERLN`	Underline

The format is:

`LATCHATT = attribute = value attribute = value ...`

So a typical `LATCHATT` might look like:

`LATCHATT = HILIGHT = 1 BLINK = 5 UNDERLN = 4 REVERSE = 7`

If the equal sign and value are missing, the attribute is given the value (binary) 1.

Most ANSI terminals use latch attributes, and the codes are fairly standardized. To determine which attributes are supported and how attributes can be combined, if at all, examine the `SGR` entry that usually follows the `LATCHATT` entry. Some ANSI terminals support color, either foreground only or foreground and background. The sequences for color are far less standard.

Terminal manuals often describe the sequence as “set graphics rendition.” A common description reads:

```
ESC [ p1 ; p2 ; ... m
```

where p n=	0 for normal

1 for bold
5 for blink

Thus ESC [0 m is normal, ESC [1 m is bold, ESC [1 ;5 m is bold and blinking. Often, setting an attribute does not remove others, so it is best to reset to normal first, using ESC [0; 1 m for bold, ESC[0;1;5m for blinking bold, and so on. The coding in the video file is as follows:

```
LATCHATT = HILIGHT = 1 BLINK = 5 UNDERLN = 4 REVERSE = 7
SGR = ESC [ 0 %9(%t ; %c %; %) m
```

The meaning of the above SGR sequence is as follows. The sequence is passed 11 parameters, each 0 (if the attribute is not to be set) or the character in the LATCHATT list. First, ESC [0 is output. The %t test, repeated 9 times, causes the zero parameters to be skipped. A non-zero parameter causes a semicolon and the parameter to be output. Finally, the character m is output. If the normal attribute is wanted, all parameters are 0, and the output sequence is ESC [0 m. For underline SGR is ESC [0 ; 4 m. If highlighted, blinking, and reverse video are desired, the output is

```
ESC [ 0; 7 ; 5 ;1 m.
```

Unable to combine attributes

Some terminals (or emulators) do not accept the method of combining attributes used above. In that case, one sequence followed by the next might work—for example, ESC [1 m ESC [7m. Some terminals cannot combine attributes at all. Here are some more ANSI and near-ANSI examples:

Standard ANSI terminal

```
LATCHATT= HILIGHT=1 BLINK=5 UNDERLN=4 REVERSE=7
```

ANSI with low intensity but no highlight

```
LATCHATT= DIM=2 REVERSE=7 UNDERLN=4 BLINK=5
```

Only one attribute available:

```
LATCHATT= REVERSE=7
```

Repeat of above SGR example

```
SGR = ESC [ 0 %9(%t ; %c %; %) m
```

Attributes cannot be combined

```
SGR = ESC [ 0 m %9(%t  ESC [ %c m %; %)
```

Skip parameters that are always 0

```
SGR = %u ESC [ 0 %5(%t ; %c %; %) m
```

In the next LATCHATT/SGR example explicit pushes are used to select the appropriate parameter. The second pair is the same as the first, but the attribute is treated as a boolean. The first uses the optional %?, the second omits it.

```
LATCHATT = DIM = 2
SGR = ESC [ m %? %p5 %t ESC [ 2 m %;
```

```
LATCHATT = DIM
SGR = ESC [ m %t ESC [ 2 m %;
```

The following is suitable for terminals that support all attributes but cannot combine them. It selects one attribute giving preference to REVERSE, UNDERLN, BLINK, and HILIGHT in that order. It uses a complicated "if-then-elseif-elseif-elseif" structure. Automatic parameter sequencing cannot be relied on, so explicit parameter pushes are used.

```
LATCHATT = HILIGHT BLINK UNDERLN REVERSE
SGR = ESC [ %p3 %t 7 %e %p2 %t 4 %e %p4 %t 5 %e \
      %p6 %t 1 %; %; %; %; m
```

Bit-mapped attributes

Some terminals use bit-mapped attributes. Terminal manuals are not usually explicit on this. Often they use tables like that described in Table 7-5.

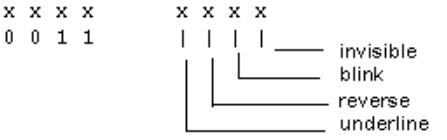
Table 7-5 Bit-mapped attributes

n	Visual attribute	n	Visual attribute
0	normal	8	underline
1	invisible	9	invisible underline
2	blink	:	underline and blink
3	invisible blink	;	invisible underline and blink
4	reverse video	<	reverse and underline

Table 7-5 Bit-mapped attributes

n	Visual attribute	n	Visual attribute
5	invisible reverse	=	invisible reverse and underline
6	reverse and blink	>	reverse, underline and blink
7	invisible reverse and blink	?	invisible reverse, underline and blink

After poring over the ASCII table for a while, it becomes clear that this is bit-mapped, with the four high-order bits constant (0x30) and the four low-order bits varying, like this:



This can be coded in the video file as follows. The attributes are OR-ed with a starting value of '0' (0x30).

```
LATCHATT = BLINK = 2 REVERSE = 4 UNDERLN = 8
SGR = ESC G %'0' %9( %| %) %c
```

For Videotex terminal

Following are examples of `LATCHATT` entries that can be used with a Videotex terminal. The `LATCHATT` entries, when used with the `SGR` entry, have equivalent equivalent results. The bits are OR-ed together with a starting value of 0x40, or `@`, and the result is output as a character.

```
LATCHATT= UNDERLN=DLE BLINK=STX REVERSE=EOT HILIGHT=SP
LATCHATT= UNDERLN= ^P BLINK= ^B REVERSE= ^D HILIGHT= SP
LATCHATT= UNDERLN= 0x10 BLINK= 0x02 REVERSE= 0x04 \
          HILIGHT= 0x20

LATCHATT= UNDERLN= P BLINK= B REVERSE= D HILIGHT= `
SGR = FS G %'@' %9( %| %) %c
```

Protected modes

Some terminals that use area attributes support a single latch attribute. It is often called "protected" and is used to indicate protected areas when the terminal is operated in block mode. The following example switches between protected and unprotected modes in order to use low intensity. (Be aware that a terminal might become very slow when using the protect feature.) The `SGR` sequence depends only on the attribute being non-zero, so no value is necessary:

```
LATCHATT = DIM
SGR = ESC %?%t ) %e ( %;
```

`SGR`

A set graphics rendition sequence, in conjunction with the `LATCHATT` sequence, is passed twelve parameters.

1	standout
2	underline
3	reverse video
4	blink
5	dim (low intensity)
6	highlight (bold)
7	blank
8	protect not used, always 0
9	alternate character
10	foreground color (if available)
11	background color (if available)
12	background highlight

If an attribute is desired, the parameter passed is the character associated with the attribute, as explained below. If the attribute is not desired, the parameter passed is (binary) 0.

For example, a video file might contain this entry:

```
LATCHATT = REVERSE = 7 HILIGHT = 1 BLINK = 5 UNDERLN = 4
```

If a field is to be highlighted and underlined, the SGR sequence is passed (0, '4', 0, 0, 0, '1', 0, 0, 0). The second and sixth parameters represent underline and highlight; they are set to the corresponding values in the LATCHATT entry. The rest are zero. To make the field reverse video and blinking, SGR is passed (0, 0, '7', '5', 0, 0, 0, 0, 0) .

If no attributes are specified in the video file, Panther supports only two attributes: non-display (done in software anyway) and underline (using the underscore character).

SPXATT

Lists attributes which do not change or affect the appearance of a character cell containing a space, for example:

```
SPXATT = BOLD DIM BLANK BLINK COLOR
```

For efficiency, this entry reduces the number of characters sent to a screen. It defaults to COLOR BLANK HILIGHT DIM. To turn it off entirely, use:

```
SPXATT =
```

Status Line

Panther usually uses a line from the screen to display status text and error messages. Thus a 25-line screen (as specified in the LINES keyword) has 24 lines for the screen, and one for messages. This use of a normal screen line for messages is the default.

Terminals that use a separate status line can use different attributes on the status line than on the screen itself. Panther provides some support for this ability; for very complicated status lines, you must write a routine and install it with sm_install using the STAT_FUNC function type. For more information on installing a status line function, refer to “Status Line Function” on page 44-45 in *Application Development Guide*.

OMSG and CMSG

Opens and closes the status line. Use the OMSG entry to open the status line, and CMSG to close it. These entries are used primarily for those terminals that have a special status line that cannot be addressed by normal cursor positioning. All text between these sequences appears on the status line. No assumption is made about clearing the line; Panther always writes blanks to the end of the line.

```
OMSG = ESC f
CMSG = CR ESC g
```

If the OMSG entry is present in your video file, Panther uses all the lines specified in the LINES entry for screens.

MSGATT

Lists the attributes available on the status line. This keyword takes a list of flags:

AREAATT	All area attributes can be used
BLINK	Blink available
DIM	Dim (low intensity) available
HILIGHT	Highlight (bold) available
LATCHATT	All latch attributes can be used
NONE	No attributes on status line
ONSCREEN	Area attributes take a screen position
REVERSE	Reverse video available
UNDERLN	Underline available

The attribute for the status line is specified as either a latch (LATCHATT) or area (AREAATT) attribute, and the sequence to set it must is given in the SGR or ASGR keyword. For example, if REVERSE is listed in MSGATT and REVERSE is a latch attribute, then SGR sets it. Attributes that appear in MSGATT and do not appear in either LATCHATT or AREAATT are ignored.

In order for Panther to determine the correct length of a line, it is important to know whether area attributes are onscreen or not. It is not uncommon for area attributes to be non-embedded on the screen but embedded on the status line. Use the ONSCREEN flag in MSGATT to inform Panther of this condition.

```
LATCHATT = DIM
AREAATT = REVERSE UNDERLN BLINK
MSGATT = REVERSE UNDERLN BLINK ONSCREEN
MSGATT = AREAATT ONSCREEN
```

The two MSGATT entries are equivalent. They show a case where only area attributes are available on the status line and they take a screen position. The area attributes in the normal screen area do not.

Graphics and International Character Support

Panther has support for eight-bit ASCII codes as well as any graphics that the terminal can support in text mode. Bitmapped graphics are not supported.

GRAPH

Maps internal numbers to output sequences, similar to the key translation files which provide mapping from character sequences to internal numbers. The only character value that can not be sent is 0.

Some terminals have a special compose key, active in eight-bit mode. Generally, you would press the compose key followed by one or two more keys, generating a character in the range 0xa0 to 0xff. Panther can process such characters as normal display characters, with no special treatment in the video file.

Other terminals have special keys that produce sequences representing special characters. The key translation file can be used to map such sequences to single values in the range 0xa0 to 0xfe. (Refer to the *Developer's Guide* for a way to use values outside that range.) The video file can then specify how these values are output to the terminal.

Often, to display graphics characters, a terminal must be told to “shift” to an alternate character set (in reality, to address a different character ROM). The video file's GRAPH entries tell which alternate set to use for each graphics character, and how to shift to it. Whenever Panther is required to display a character, it looks in the GRAPH table for that character (refer to the description on MODE0 through MODE6 for information on what happens). If it is not there, the character is sent to the terminal unchanged.

GRTYPE

Provides a convenient shortcut for certain common graphics sets, each denoted by another keyword. The GRTYPE keywords can be combined. The format is:

GRTYPE = [GRTYPE] *keyword*...

The GRTYPE *keyword* can be any of the following:

ALL	0xa0 through 0xfe
EXTENDED	same as ALL
PC	0x01 through 0x1f and 0x80 through 0xff
CONTROL	0x01 through 0x1f, and 0x7f
C1	0x80 through 0x9f, plus 0xff

An entry in the GRAPH table is made for each character in the indicated range, with mode 0. If the mode is not 0, you must construct the GRAPH table by hand.

MODE0 through MODE6

Panther supports up to three alternate character sets. The sequences that switch among character sets are listed below.

MODES0 through MODE3 are locking shifts. All characters following are shifted, until a different shift sequence is sent.

MODES4 through MODE6 are non-locking or single shifts, which apply only to the next character.

You may need to use the INIT entry to load the character sets you want for access by the mode changes.

MODE0	switch to standard character set
MODE1	alternate set 1
MODE2	alternate set 2
MODE3	alternate set 3
MODE4	...
MODE5	
MODE6	

Different modes can be used to support foreign characters, currency symbols, graphics, and so on. Panther makes no assumption as to whether the mode changing sequences latch to the alternate character set or not. To output a

character in alternate set 2, Panther first outputs the sequence defined by MODE2, then a character, and finally the sequence defined by MODE0 (which may be empty, if the others are all non-locking). Here are three examples:

```
MODE0 = SI
MODE1 = SO
MODE2 = ESC n
MODE3 = ESC o
```

Sample of ANSI standard

```
MODE0 = ESC [ 10 m
MODE1 = ESC [ 11 m
MODE2 = ESC [ 12 m
MODE3 = ESC [ 13 m
MODE0 =

MODE1 = SS1
MODE2 = SS2
```

Any of the MODE*n* strings may also contain a list of attributes. When a character in that mode is displayed, that attribute is added to whatever attribute is already in effect. On some terminals, like the HP, only an attribute is required. For example:

```
MODE4 = ACS
```

This forces all mode 4 characters to be displayed using the alternate character set.

Any character in the range 0x01 to 0xff can be mapped to an alternate character set by use of the keyword GRAPH. The value of GRAPH is a list of equations. The left side of each equation is the character to be mapped; the right side is the number of the character set (0, 1, 2, 3), followed by the character to be output. Any character not so mapped is output as itself. For example, suppose that 0x90 =1 d appears in the GRAPH list. First the sequence listed for MODE1 is sent, then the letter d, and then the sequence listed for MODE0.

In the following example, 0x81 is output as SO / SI, 0xb2 as SO 2 SI, and 0x82 as ESC o a SI. LF, BEL and CR are output as a space, and all other characters are output without change. This output processing applies to all data coming from Panther. No translation is made for direct calls to printf, putchar, and so on. Thus \n and \r will still work correctly in printf, and putchar (BEL) still rings the terminal bell.

```
MODE0 = SI
```

```
MODE1 = SO
MODE2 = ESC
nMODE3 = ESC o
GRAPH = 0x81 = 1 / 0xb2 = 1 2 0x82 = 3 a LF = 0 SP\
        BEL = 0 SP CR = 0 SP
```

Recommendations

For efficiency, use single shifts to obtain accented letters, currency symbols, and other characters that appear mixed in with unshifted characters. Graphics characters, especially for borders, are good candidates for a locking shift.

It is possible, though not recommended, to map the usual display characters to alternates. For example, `GRAPH = y = 0 z` causes the y key to display as `z`. Graphics characters are non-portable across different displays, unless care is taken to ensure that the same characters are used on the left-hand side for similar graphics, and only for a common subset of the different graphics available.

Borders and Line Drawing

The characters constituting the border and line drawing styles can be specified in the video file.

`BORDER`

Specifies alternate borders. If fewer than 10 are given, the default borders (listed below) are used to complete the set. They are numbered 0 to 9. When you create screens with the screen editor, you can select from these border styles. The `BORDER` entry is portable across different platforms because Panther saves a border as a style number in the screen file. Style 1 is the default.

Border Styles

```
0.                                  1.
IIIII                               _____
I   I                               |   |
IIIII                               _____

2.                                  3.
+++++                               =====
+   +                               |   |
+++++                               =====
```

```
4.
%%%%%
%   %
%%%%%

5.
.....
:   :
.....

6.
*****
*   *
*****

7.
\\\\\
\   \
\\\\\

8.
/////
/   /
/////

9.
#####
#   #
#####
```

The data for BORDER is a list of 8 characters per border, in the order: upper left corner, top, upper right corner, left side, right side, lower left corner, bottom, lower right corner. The default border set is:

```
BORDER =   SP  SP  SP  SP  SP  SP  SP  SP  \
           SP  _   SP  |   |   |   _   |   \
           +   +   +   +   +   +   +   +   \
           SP  =   SP  |   |   SP  =   SP  \
           %   %   %   %   %   %   %   %   \
           .   .   .   :   :   :   .   :   \
           *   *   *   *   *   *   *   *   \
           \   \   \   \   \   \   \   \   \
           /   /   /   /   /   /   /   /   \
          #   #   #   #   #   #   #   #
```

In addition, attributes can be specified for each set of border characters. For example:

```
BORDER = SP  SP  ...  SP  REVERSE \
         ;   A   ...  +   ACS \
```

If there is a GRAPH entry in the video file, you can use the graphics character set of the terminal for borders. Choose some numbers to represent the various border parts. The GRAPH option can be used to map these numbers to a graphics character set. The numbers chosen are arbitrary, except that they should not conflict with ordinary display characters. Even if the extended 8-bit character set is used, there are unused values in the ranges 0x01 to 0x1f and 0x80 to 0x9f.

Boxes

BOX

Ten different sets of line draw characters can be specified when you are using the screen editor. The BOX entry lists either five or thirteen characters per set. If only five characters are specified the remaining eight are taken from the corresponding BORDER set.

Although the format in the video file is similar, Panther uses BOX and BORDER differently. While BORDER entries are portable across platforms, Panther saves line drawing as display data. To ensure portability, avoid assigning graphic characters to the BOX keyword. Instead, use characters that are displayable on all terminals.

BRDATT

Use a BRDATT entry to limit the attributes available in the border. Normally HILIGHT (or DIM) and REVERSE are used; however, if your terminal uses onscreen attributes or can accommodate only a few attributes per line, it may be better to prohibit attributes in borders—use the entry BRDATT = NONE.

The flags used in MSGATT can also be used with BRDATT; however, the only attributes available are HILIGHT, DIM, and REVERSE.

Indicators

ARROWS

Used to indicate the presence of offscreen data in shifting/scrolling fields. You can define these indicators to be any characters you wish. The default characters are:

```
<, >, X for shifting
^, v, X for scrolling
```

The character X is used when two shifting/scrolling fields are next to each other; it represents a combination of both < and >.

```
ARROWS = < > X ^ v X
```

Shift/scroll indicators are black on screens with background colors of white, yellow, or cyan. They are white on all other screen background colors. You cannot alter indicator colors.

`BELL`

If present, is transmitted by the library function `sm_bel` to give a visible alarm. Normally, the function rings the terminal's bell. The `BELL` sequence can sometimes be found in the `termcap` file under `vb`.

`CBDSEL` and `CBSEL`

Selection (`CBSEL`) and deselection (`CBDSEL`) characters for groups. If there are no entries for `CBSEL` and `CBDSEL` in your video file, the internal defaults are X for `CBSEL` and a blank for `CBDSEL`. For radio button or checklist widgets, these characters are used to indicate which fields are selected and which are not. You can add these entries to the video file to override the defaults. For example:

```
CBSEL  =  y
CBDSEL =  n
```

As a result, Panther uses a `y` to indicate a selected occurrence; NL deselects the occurrence, and Panther inserts an `n` in the box.

`MARKCHAR`

Defines the character used to check menu items. The default character is `X`. For example, the following example specifies the square root symbol (/) as the mark character. This variable is optional for supporting those character-mode applications that use menu bars and is most useful in video files that support reasonable graphical characters.

```
MARKCHAR = 0xFB
```

`OPTMNUIND`

Defines the character used to indicate option menus. The default character is `v`. This variable is optional for character-mode applications.

`SLIDER`

Defines the characters used in character-mode scroll bars. Eight characters are defined, where the first set of four characters define the slider characters used in horizontal scroll bars, and the second set contains the slider characters used in vertical scroll bars:

```
SLIDER = left-arrow right-arrow bar thumb \
         up-arrow down-arrow bar thumb
```

For example, these characters are defined in `vt100vid`:

```
SLIDER = < > = # ^ v / #
```

You can use hex values of graphics characters for those terminals that support them. For example:

```
SLIDER = 0x11 0x10 0xb0 0xb2 0x1e 0x1f 0xb0 0xb2
```

`SUBMNSTR`
: Defines the indicator for submenu options. This indicator appears on menu bar options indicating that the item invokes a submenu. The default string is an ellipsis (...). This variable is optional for supporting those character-mode applications that use menu bars and is most useful in video files that support reasonable graphical characters.

Drivers

`MOUSEDRIVER`
: Enables the mouse in Panther applications. Supported mouse types on UNIX are `AT`, `SCO`, `Jterm`, and `xterm`.

Miscellaneous

`COMPRESS`
: Implements data compression for J*term* users. Use the following entry:

```
COMPRESS = JTERM
```

`CURPOS`
: Specifies the time-out delay, in tenths of a second. The `CURPOS` entry causes the current cursor position to be displayed on the status line at the specified intervals:

`CURPOS = 1`
: Updates display every 0.1 second (use on fast systems)

`CURPOS = 3`
: Updates every 0.3 second (reasonable for most)

`CURPOS = 7`
: Updates every 0.7 second at low baud rates

`CURPOS = 0`
: No display, same as omitting keyword.

When possible, Panther uses non-blocking keyboard reads. If no key is received within a specified time, the cursor position display is updated. This allows fast typists to type at full speed; when the typist pauses, the cursor position display is updated.

The delay depends on the baud rate and your terminal. If there is no non-blocking read, a non-zero value of `CURPOS` enables the display and zero disables it. If the terminal has its own display, omit `CURPOS`.

Sample Video File

This sample video file is for a basic ANSI terminal (like a VT100) It contains the basic capabilities, plus control sequences to erase a line and to apply the reverse video, underlined, blinking, and highlighted visual attributes. The entries for `CUP` and `SGR` are more complicated because they require additional parameters at runtime.

```
# Display size (these are actually the default # values)
LINES = 24
COLMS = 80

# Erase whole screen and single line
ED = ESC [ 2 J
EL = ESC [ 0 K

# Position cursor
CUP = ESC [ %i %d ; %d H

# Standard ANSI attributes, four available
LATCHATT = REVERSE = 7 UNDERLN = 4 BLINK = 5 HILIGHT = 1
SGR = ESC [ 0 %u %5(%t ; %c %; %) m
```

CHAPTER

8 Setup File Utilities

This chapter describes command-line utilities that let you convert Panther setup files from source to binary format. Utilities are listed in alphabetical order; descriptions are organized into the following components, as applicable:

- Utility name and brief description.
- Syntax line and argument descriptions.
- Description of the utility.
- Possible errors and corrective action.

key2bin

Converts an ASCII key translation file into binary format

```
key2bin [-pv] [-e ext] keyFile ...
```

Arguments

`-p`
: Places the binary files in the same directories as the key translation files.

`-v`
: Lists the name of each key translation file as it is processed.

`-e ext`
: Replaces default `bin` extension with the specified extension (*`ext`*) on the output file.

`keyFile`
: The name of an ASCII key translation file; more than one key translation file can be included. By convention, the key translation filename is an abbreviation of the terminal's name plus keys. The tag keys helps identify the file as a key translation file; for example, `vt100keys` is the key translation file for a VT100. `key2bin` converts an ASCII key translation file that you have edited or created into binary format for use by applications using the Panther library.

Description

`key2bin` first tries to open its key translation file with the exact name you enter on the command line; if that fails, `key2bin` appends keys to the name and tries again. The output file is given the name of the successfully opened key translation file plus the default extension (`.bin`).

To make a key translation file memory-resident, run the `bin2c` utility on the binary file, compile the resulting program source file, and link it with your application. For a complete description of how to make configuration files memory-resident, refer to "Including Memory-Resident Components," on page 42-8 in *Application Development Guide*.

Errors

The following table describes possible errors, their causes, and the corrective action to take.

Cannot create ‘Error writing’	
Cause	An output file could not be created because of lack of permission or perhaps disk space.
Action	Correct the file system problem and retry the operation.
Duplicate key definition in line:\n '%s'	
Cause	The same character-sequence was assigned to more than one logical key in the specified key translation file.
Action	Edit the ASCII file and assign a unique character-sequence to key or keys in question. Then run `key2bin` again.
Neither '%s' nor '%s' found.	
Cause	A key translation file was missing or unreadable.
Action	Check the spelling, presence, and permissions of the file in question.
Unable to allocate memory	
Cause	`key2bin` could not allocate enough memory for its needs.
Action	None.
No key definitions in file '%s'	
Cause	Warning only. The key translation file was empty or contained only comments.
Action	None.
Unknown mnemonic in line: '%s'	
Cause	The line printed in the message does not begin with a logical key mnemonic.
Action	Refer to `smkeys.h` for a list of mnemonics, and correct the input.

Extra characters in sequence neglected, in line: '%s'	
Cause	The key sequence is longer than the maximum of six characters.
Action	Correct the input.
At least one file name is required.	
Cause	One or more options was specified but no key translation file names were given.
Action	Specify file name(s).

var2bin

Converts ASCII setup files to binary format

```
var2bin [-pv] [-e ext] sourceFile...
```

`-p`
: Places the binary output file in same directory as the input file.

`-v`
: Lists the name of each input file as it is processed.

`-e ext`
: Replaces the default bin extension on the output file with the specified extension (*ext*).

Description

The name of an ASCII setup file; you can specify more than one input file.

The output of `var2bin` is a binary file having the name of the file you have specified, with a default extension of bin. You designate this output file to be used as a setup file in either the `SMVARS` or `SMSETUP` variables, or in the system environment.

Errors

The following table describes possible errors, their cause, and the corrective action to take.

%s is an invalid name.	
Cause	The indicated line did not begin with a valid variable name.
Action	Refer to Chapter 2, “Application Variables,” for lists of variable names. Correct the ASCII input file, and run `var2bin` again.
At least one file name is required.	
Cause	You have failed to give an input filename.
Action	Retype the command, supplying the ASCII setup filename.

Error opening %s.	
Cause	An input file was missing or unreadable.
Action	Check the spelling, presence, and permissions of the file in question.
Missing '='.	
Cause	An input line did not contain an equal sign after the variable name.
Action	Correct the ASCII input file by inserting the equal sign and run `var2bin` again.
Unable to allocate memory.	
Cause	The utility could not allocate enough memory for its needs.
Action	None.
%s is an invalid parameter.	
Cause	An option in the input is misspelled or misplaced, or conflicts with an earlier option.
Action	Check the valid options listed in Chapter 2, “Application Variables.”. Correct the ASCII input file and run `var2bin` again.

vid2bin

Converts a video file to binary

```
vid2bin [-pv] [-e ext] vidFile...
```

Arguments

`-p`
: Places the binary output file in same directory as input file.

`-v`
: Lists the name of each video file as it is converted.

`-e ext`
: Replaces the default bin extension with the specified extension (`ext`) on the output file. `vid2bin` converts ASCII video files to binary format. The ASCII video files distributed with Panther have been complied and their binary versions reside in the `config` directory.

`vidFile`
: The name of an ASCII video file. Customarily, it is an abbreviation (mnemonic) of the terminal name followed by the suffix `vid`—for example `sunvid` for a terminal, or `colvid` for a color monitor.

Description

`vid2bin` searches for `vidFile`, first trying the mnemonic, then the mnemonic followed by `vid`. The output file gets the same name as the input file, with the extension bin or the extension (`ext`) that is specified by the -e option.

To make a video file memory-resident, run the `bin2c` utility on the binary output, compile the resulting program source file, and link it with your application. For a complete description of how to make configuration files memory-resident, refer to "Including Memory-Resident Components," on page 42-8 in *Application Development Guide*.

`vid2bin` checks for errors like missing, misspelled, and superfluous keywords, but not for duplicated or conflicting entries. If errors are encountered, up to ten error messages can be displayed; no output file is created.

Errors

The following table describes possible errors, their causes, and the corrective action to take.

A cursor positioning sequence is required.
An erase display sequence is required.

Cause	Both entries (CUP and ED) are required in video files.
Action	Determine what your terminal uses to perform these two operations, and enter them in the ASCII video file; then run `vid2bin` again.

Invalid entry: '%s'.Entry missing '=': '%s'.

Cause	An input line does not begin with a valid video keyword. An input line does not include and an equal sign.
Action	Correct the ASCII file and run `vid2bin` again. Be sure that backslashes are placed at the end of lines that continue onto the next line.

Invalid attribute list : '%s'.
Invalid border information (%s):'%s'.
Invalid box information (%s): 's%'
Invalid color specification : '%s'.
Invalid cursor flags specification : '%s'.
Invalid graphics character specification (%s):'%s'.
Invalid graphics type : '%s'.
Invalid label parameter : '%s'.%s
Invalid numeric parameter : '%s'

Cause	There is a misspelled or misplaced keyword in the specified input line.
Action	Correct the ASCII video file, and run `vid2bin` again.

Neither %s nor %s found.

Cause	The video file was missing or unreadable.
Action	Check the spelling, presence, and permissions of the file in question.

Unable to allocate memory.	
Cause	The utility could not allocate enough memory for its needs.
Action	None.

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Z