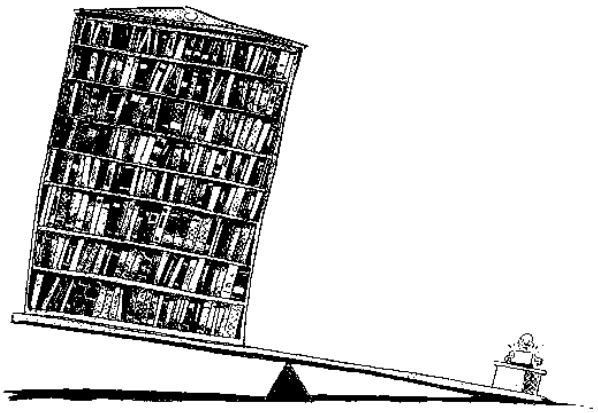
KE Texpress



Utilities Guide

KE Software Pty Ltd

Contents

Chapter 1

Introduction

Overview

This manual describes numerous utility programs provided within the KE Texpress system. Most of these facilities are invoked directly from the Unix shell, however a few are selected via the database Admin mode and require the user to have the appropriate Admin database privileges. Also, some of the programs invoked from the shell may be used only by the DBA.

Using this Guide

The **texcreate** program is used to create a new database. The user executing the command becomes the DBA for the database. Programs **texrename** and **texcopy** are used to move and copy a database respectively. The DBA of a database may be changed using the **texchdba** program, which can be performed only by the Unix super user. A database can be removed using the **texdelete** command. A list of all database names and DBAs may be obtained using the **texlist** command. These facilities are described in detail in Chapter 2.

The KE Texpress auditing facility is described in Chapter 3. Auditing allows an administrator to maintain a log of all significant operations performed on the database. Each log entry indicates the user, the type of operation and the date and time it was performed. Audit control facilities are available on the Audit menu, which is selected from the Administrator screen. The Audit menu contains options for enabling and disabling auditing, flushing information from the audit log and displaying the size of the audit information currently maintained. The program, **texaudit**, which is invoked from the Unix shell, can be used to generate reports from the audit information. These reports can incorporate costing details for specific tasks and connection times.

Chapter 4 describes a facility for easy development of front end menus. Front-end menus can provide swift access to a variety of databases and facilities.

In Chapter 5 general purpose label printing facilities are described. These can be used to print a wide variety of label formats on printers which accept the Postscript language.

Command Notation

Commands run within KE Texpress can be performed from pull-down menus, or via keyboard accelerators (characters). Often, a command can be performed in several ways. For uniformity and ease of use, only the pull-down menu option is described in the guide.

The *Perform Query* command can be run by selecting the *Query* option on the *Function* pull-down menu in *Query* mode. The guide would describe it as:

```
Perform Query ([Query] Function  ⇒ Query)
```

where the description of the command is followed by the pull-down menu option in brackets. The mode is contained in square brackets, e.g. [**Query**] and the pull-down menu name **Function** is followed by ⇒ and the command **Query** exactly as listed on the pull-down menu.

Alternatively, the Perform Query command can be performed from the keyboard, by holding down the **Control** key and typing **Y**. In the guide, the keystrokes are described as **Ctrl+Y**.

Many of the common commands may be invoked directly by a particular function key (provided the terminal supports function keys). Throughout KE Texpress (and the screen images displayed in this document), function keys are represented by the letter **F** followed by the function key number. Thus the command generated by pressing function key number one is represented as **F1**.

Other special symbols used in this document are:

ESC	The **escape** key.
↵	The **return** or **enter** key.
DEL	The **delete** or **interrupt** key (used to interrupt an operation).
Space	The **space bar**.
Backspace	The **backspace** key.
Tab	The **tab** key.
←	The **left arrow** key.
→	The **right arrow** key.
↑	The **up arrow** key.
↓	The **down arrow** key.

Chapter 2

Management Utilities

Overview

This chapter covers several database management utility programs. Each of these programs is invoked directly from the shell. Some require database administrator privilege. They include how to create, rename, copy, delete and list databases, as well as change theDBA of databases.

Creating a Database -texcreate

A program called **texcreate** is used to create a new KE Texpress database.

It is invoked from the Unix shell in the format:

```
texcreate dbname [dbname ...]
```

where ***dbname*** is to become the name of the new database. The database name must be unique among databases on the current system. The user issuing the command must be allowed to be a Database Administrator (see following). All programs and files needed for the new database are created silently. The user issuing the command automatically becomes the DBA of the new database.

Typically databases are created in dedicated directories which administrators and users are not required to know. However, databases can be created in a specific directory by defining one or more database options. These options specify the Unix directory paths for database files. For a complete description of database path options refer to the KE Texpress User Guide or KE Texpress Design Guide.

Database Administrators

Eligibility of users to become DBA's is determined by the existence or non-existence of login names (one per line) in the following files:

```
~texpress/etc/dba/allow
~texpress/etc/dba/deny
```

These files may only be edited by the KE Texpress administrator (Unix login account texpress). The following rules describe the order of evaluation for DBA eligibility:

- If a users name is in the deny file then that person is not permitted to be a DBA;
- Otherwise, if the allow file contains one or more user names then a particular users' name must be in the file for that person to be a DBA;
- Otherwise, any person may be a DBA.

Before commencing the upgrade procedure, as user texpress you should edit the allow and deny files as appropriate.

For KE Texpress the login name texpress must be the only user who is a member of Unix group texpress. The Unix account texpress should only be used for administration of the KE Texpress software and generally not as a DBA.

Renaming a Database -texrename

A database can be renamed using the program **texrename**. It is invoked from the Unix shell in the format:

```
texrename fromdbname todbname
```

The new database name must not be longer than eleven characters and must be unique among databases on the current system. The user issuing the command must be the DBA of the old database.

A database can be moved to a new location within the Unix directory hierarchy as it is renamed by setting the relevant database path options before the **texrename** command is invoked. Refer to the KE Texpress Design Guide for a description of database path options.

Copying a Database -texcopy

A database can be copied using the program **texcopy**. It is invoked from the shell in the format:

```
texcopy fromdbname todbname
```

The new database name must not be longer than eleven characters and must be unique among databases on the current system. The user issuing the command must have a user account at privilege level 0 for the database.

A database can be copied to a new location within the Unix directory hierarchy by setting the relevant database path options before the texcopy **command** is invoked. Refer to the KE Texpress Design Guide for a description of database path options.

Changing the Database Administrator -texchdba

The administrator of a database can be changed using the **texchdba** command. Only the Unix super user may run this command. It is invoked from the Unix shell in the format:

```
texchdba dba dbname [dbname ...]
```

where **dba** is the Unix login Id of the new database administrator. The new administrator must be a member of the Unix group KE Texpress. The program will respond with a confirmation message in the format:

```
"dbname" database
```

```
Old DBA was olddba
New DBA is dba
```

Refer to the section "Creating a Database" in this guide for information on the eligibility of users to be Database Administrators.

Removing a Database -texdelete

The command **texdelete** can be used to remove a database. This permanently deletes the database and all associated files, so this command should be used with caution. Only the DBA or Unix super user may remove a database. The command is invoked from the Unix shell in the format:

```
texdelete [-f]  dbname [dbname ...]
```

Initially a warning message is displayed, with a prompt for confirmation that the command should be continued. Choosing no aborts the remove procedure. Choosing yes causes another confirmation prompt to display:

```
Last chance before database is permanently removed. Abort?
```

Choosing no removes the database after the following messages are displayed

```
Removing "dbname" database ...
```

and

```
"dbname" database is removed
```

Pressing any key returns control to the Unix shell or if multiple database names were specified on the command line, the remove procedure continues with the next database.

If the command line option **-f** is given, then each specified database will be removed silently (if the user is the DBA) without confirmation being requested. This option should be used with caution.

Listing Databases -texlist

The program **texlist** can be used to give information on all KE Texpress databases. The program is run from the Unix shell. In its simplest form, this program is invoked by typing the command.

```
texlist
```

This lists the names of all of the KE Texpress databases in alphabetic order. The Unix user Id of the DBA for each database appears in parentheses to the right of the database name. The database names are printed in two column format.

Several options can alter the operation of this program, and are invoked by a command in the format:

```
texlist [-bdlnprs] [-uuid] [dbname ...]
```

Sections enclosed by [and] are optional. The character string, bdlnprs, represents a set of individual options which can be selected in any combination. The full list of options is as follows:

`-l` Produce a long listing of the databases. A single column output format is employed. In addition, a string of the form:

```
nnn records    mm.m%
```

is appended to the report for each database. The figure nnn, represents the number of records in the database and the figure mm.m represents the percentage of database capacity which has been used. For uninitialized databases, the record count and the percentage of capacity figures are replaced by the text:

```
Uninitialised
```

`-b` Sort the databases numerically by the total size, in bytes, of all of the files which make up each database. Databases of the same size are sorted by name.

`-d` Sort the databases alphabetically by the Unix user Id of the DBA. Databases with the same DBA are sorted by name.

`-n` Sort the databases numerically by the number of records in each. Databases with the same number of records are sorted by name.

`-p` Sort the databases numerically by the percentage of capacity used. Databases with the same percentage of capacity used are sorted by name.

`-r` Reverse the order of the listing. Thus alphabetic lists are produced in reverse alphabetic order. Numeric lists are produced in decreasing order of value.

`-s` Display the total size, in bytes, of all of the files which make up the database. The size is displayed in the form:

```
sssc
```

where sss indicates the size of the database and is qualified by the single letter, *c*. If *c* is b, the size is given in bytes. If *c* is K, the size is given in kilobytes (1024 bytes). If *c* is M, the size is given in megabytes (1048576 bytes).

`-u`*`uid`* Restrict the output to databases for which uid is the DBA. Note that this option can appear several times on the command line in which case the output is restricted to databases for which the DBA is one of the uid's specified. Note also that if uid is not a valid Unix Id then an appropriate error message is generated along with a usage message.

If two or more conflicting sorting options are specified on the command line, only the option is executed.

If, one or more database names are specified in the command, information concerning those databases only is output.

Database Access -texstatus

A program, called texstatus can be used to determine whether a user can access a database at any given time. It checks for an appropriate user account and ensures that the database is open, that there are no index rebuilds pending or being performed and that the database is currently in a state in which queries and insertions can be performed

This program has been designed to be used in the **valid** and **visible** conditions of **texmenu** and in shell scripts. It can be invoked by a command of the form:

```
texstatus [-v] [-il-y]  dbname
```

where *dbname* is the name of the database being tested. By default, it exits with an appropriate status but prints nothing.

The options are as follows:

`-v` The -v option can be used to force texstatus to print a message matching its exit status.

`-i` Test for Insertion privilege or Temporary Record Creation privilege and a database which is not set Read only.

`-y` Test for Query privilege and an appropriate valid Query form.

The exit status codes are as follows:

Status	Cause
0	The database can be accessed by the user.
1	A fatal error occurred accessing the database.
2	An invocation error occurred (invalid arguments).
3	The user does not have an account.
4	The database does not have an Insertion form.
5	was invoked with **-i** option.
6	The user does not have permission to query. This is applicable only i**fexstatus** was invoked with the **-y** option.
7	The **ins** file is not an Insertion form.
8	The Insertion form is invalid.
9	The database requires an index rebuild.
10	The database requires a Look-up-table rebuild.
11	The database requires an index rebuild (for the Key file).
12	The database is currently being checked.
13	An automatic data file update is currently being performed.
14	The Insertion form has been corrupted.
15	The database has not been initialized.
16	The database has not been configured.
17	There is no Query form (**-y** option only).
18	The **qry** file is not a Query form (**-y** option only).
19	The Query form is invalid (**-y** option only).
20	Another (maintenance) process is running preventing users from accessing the database.

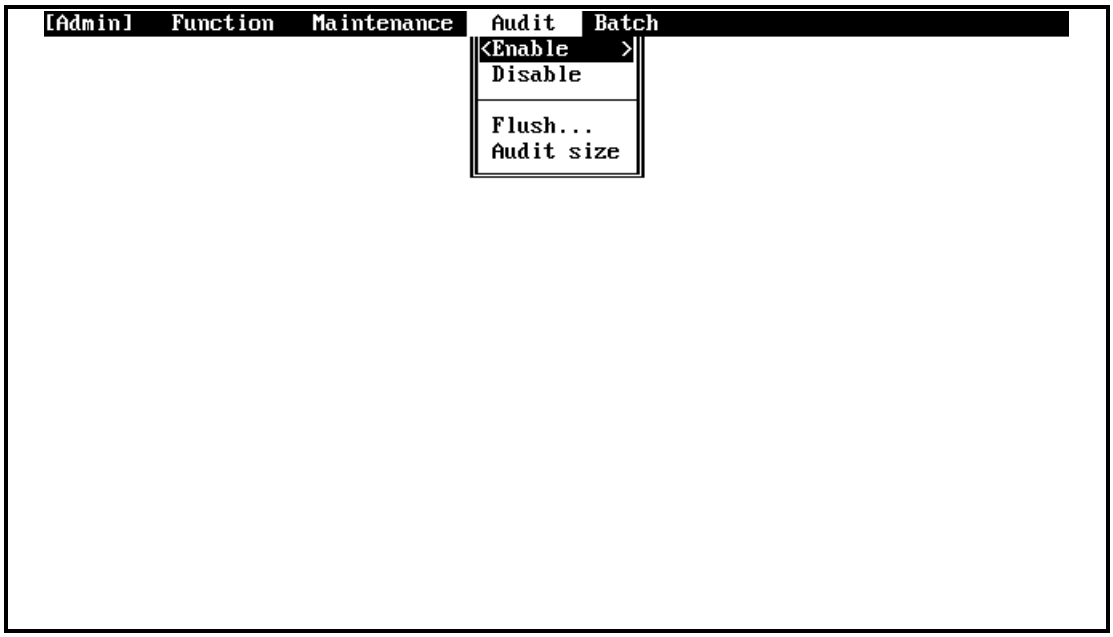
Chapter 3

Auditing

Overview

The Audit menu forms one of the pull down menus in Admin mode of the database administration program (texadmin). Its options allow an administrator to control the database auditing facilities.

If auditing is enabled, KE Texpress maintains a log of the operations performed by all database users. This operation log can be accessed using the texaudit program.



Audit menu

[Admin] Audit⇒ Enable

This command allows an administrator to enable the KE Texpress operation auditing facilities. The message:

```
Auditing enabled
```

is displayed as confirmation.

[Admin] Audit⇒ Disable

This command allows an administrator to disable the KE Texpress operation auditing facilities. This does not discard any audit information but merely disables the collection of operation information until auditing is re-enabled. The message:

```
Auditing disabled
```

is displayed as confirmation.

[Admin] Audit⇒ Flush

Large amounts of audit information can be obtained during heavy periods of database usage. This command is used to flush (or discard) audit information no longer required. The prompt:

```
Enter cut-off date:
```

is displayed and a cut-off date may be entered. The accepted date format is dd/mm/yy where dd is the day, mm a numeric month and yy a two digit year. All auditing information collected before the entered date is discarded. If a . is entered all auditing information, except that for the current day, is discarded. The message:

```
Flushing audit file ...
```

is displayed while the flush is in progress. The new size of the audit file is then displayed.

Admin [Audit] Audit size

This command allows an administrator to determine the disk space consumed by the audit information. When selected, a message of the form:

```
The audit information consumes nnn bytes
```

is displayed where nnn represents the size of the audit information in bytes. If this size is greater than 1024 bytes, the figure is given in kilobytes. If the size is greater than 1048576 bytes, the figure is given in megabytes.

Audit Reports

The program, texaudit, allows a database administrator to generate a summary of some or all of the operations performed on that database while auditing has been enabled. This program must be run from the Unix shell or command interpreter. In its simplest form, it is invoked by typing the command line:

```
texaudit dbname
```

This results in output with the heading:

```
User Id  Date     Time  Units C/U Total Operation   Key
-------  ----     ----  ----- --- ----- ---------   ---
```

Below this is a list of all of the operations performed on the database, dbname. Each entry in the list contains the Unix user Id of the user who performed the operation, the date and time at which the operation was performed and a word or phrase indicating the particular operation. For operations involving Key values or records which contain a Key value, the appropriate Key value is appended to the entry.

Each operation also has an associated unit weight (Units) and a cost per unit (C/U). These values and the total (Total = Units * C/U) are listed for each operation. Displayed at the end of the report is a statement of the Total cost of all listed operations (the sum of all the values in the Total column).

The unit weight value of each operation is 1 with the exception of database query where it is equal to the number of matching records and terminate session where it is equal to the number of seconds since the session commenced. This value cannot be changed. The default cost per unit value of each operation is as listed in the following table. This value can be changed by the DBA by editing the text file used by the database (see the KE Texpress Interface Reference Manual).

Operation	Description	C/U
backup	Backup database	0
badlog	Invalid user login attempt	0
close	Close database	0
clrkey	Clear Key value	0
config	Configure database	0
delete	Delete record	10
deltemp	Delete temporary record	5

disable	Disable auditing	0
display	Display matching record	1
editd	Destructively edit record	20
editnd	Non-destructively edit record	20
edittemp	Edit temporary record	10
enable	Enable auditing	0
end	Terminate session	1
flush	Flush audit file	0
init	Initialize database	0
insert	Insert record	20
insform	Insertion form edit	0
instemp	Insert temporary record	10
load	Automatically load data	0
open	Open database	0
qryform	Query form edit	0
query	Query database	1
recover	Recover after system crash	0
reindex	Automatic reindex	0
restore	Restore database from backup	0
start	Commence session	0
totemp	Move record to temporary file	5
updindex	Delayed index update	0

There are several options which can alter the operation of audit. In general, this program can be invoked by a command similar to:

```
texaudit [-h] [-uuid] [-fdd/mm/yy] [-tdd/mm/yy] [-oop] [-
kkey] dbname
```

Sections enclosed by [and] are optional. If two or more conflicting date options are specified on the command line, all but the last of these are ignored. The full list of options is as follows:

`-h` Invoking the auditing program with the command line:

```
texaudit -h
```

displays a list of all possible command operations.

`-uuid` Restrict the output to operations performed by uid, where uid is a Unix user Id. This option can appear several times on the command line, in which case the output is restricted to operations performed by one of the uids specified. If uid is not a valid Unix user Id an error message is generated along with a usage message.

`-fdd/mm/yy` Restrict the output to operations performed on or after the date, dd/mm/yy.

`-tdd/mm/yy` Restrict the output to operations performed on or before the date, dd/mm/yy.

`-oop` Restrict the output so that only the operation, op, is selected. The op can be one of the names listed in the operation table generated by the -h option to texaudit. Only leading letters of these options need be specified. If, however, the leading letters specified match more than one option, then all matching options are considered to be selected. This option can appear several times on the command line, indicating simultaneous selection of the specified options.

`-kkey` Restrict the output to operations performed on the Key value, key, or on the record containing the Key value, key. This option can also appear several times on the command line in which case the output is restricted to operations performed on any of the Key values specified.

Chapter 4

Front End Menus

Overview

A facility for developing front-end menus is provided with the KE Texpress software. The front-end menu displays program options and accepts simple selection commands to invoke the programs associated with each option.

Invocation

The front end menu program is called texmenu and is invoked from the shell with a command of the format:

```
texmenu [-anvw] [-e [-d n]] -mmenuname -rn file file ...
```

By default, the first menu displayed is the first menu listed in the file. Command options are:

-a	Align a series of centred menu options so that they are centred as a block.
-n	Automatically number the menu options and allow them to be selected by their corresponding number.
-v	By default, menu options are validated each time a menu is displayed. This option will ensure validation is performed only the first time a menu is displayed.
-w	Suppress warnings about the menufile.
-e	When an option is selected immediately execute the associated command, rather than waiting for ↵ to be typed.
-d*n*	Delay n seconds before executing the command. This option may used in conjunction with the -e option. It allows the user to see the menu option highlighted before the command is executed.
-m*menuname*	Rather then commencing with the first menu listed in the first file, start with the menu called *menuname*.
-r*n*	Forces the recalculation of all valid and visible conditions every **n** seconds

Menu Description File

The menu description file contains the definition of one or more menus. By default, the first such menu in the file is the first menu displayed.

Menu Name and Body

Each menu description has an associated name. The body of the menu is enclosed by curly brackets and contains one or more menu components whose structure is described in the following sections. For example:

```
menu "main"
{
        header
        {
                ...
        }
        option
        {
                ...
        }
        ...
}

menu "sub"
{
        ...
}
```

outlines a menu description file containing two menus, namely main and sub.

The menu body consists of optional **origin** and **dimension** statements followed by one or more **header**, **option** or **escape** components. Each component commences with the component type key word, followed by the component body enclosed in curly brackets. The component body consists of one or more assignment statements to attributes which are interpreted to provide the menu functionality. Each statement must be terminated with a semi-colon.

Origin and Dimension

The first lines of the menu body may contain **origin** and/or **dimension** statements.

origin

The y and x coordinates (line and column numbers), separated by a comma and enclosed by parentheses. The menu is positioned on the screen at these coordinates.

This statement is optional and if not specified the menu origin is the top left corner of the screen (0, 0).

dimension

The number of lines and columns in the menu, separated by a comma and enclosed by parentheses.

This statement is optional and if not specified the menu dimension extends from the origin to the bottom right corner of the screen.

An example menu body containing origin and dimension statements is:

```
menu "main"
{
        origin = (1, 0);
        dimension = (23, 40);
        header
        {
                ...
        }
        option
        {
                ...
        }
        ...
}
```

Header Component

A header component of a menu defines descriptive text which is displayed but does not have any associated command. The body of a header defines two attributes, namely:

position = (y,x);

The y and x coordinates (line and column numbers) separated by a comma and enclosed by parentheses. Either coordinate may be the key word **centre** which means the title is to be centred in that direction.

title = "*Title*";

The text of the title to be displayed at the designated position.

An example header component is:

```
header
{
        position = (2, centre);
        title = "Me nu Title";
}
```

which would display the title centred on the second line of the terminal screen.

Option Component

An option component of a menu defines a position and title in the same format as that for a header component. As well, the following attributes may be defined:

selection ="Aa^A";

A character string defining the keyboard characters which can be used to select the option.

action ="texforms contacts";

The command or commands to be performed when the options is selected. Generally if a complicated set of commands is necessary then the action executes the appropriate shell script.

message = "Commencing contacts ..."

A message to be displayed while the command is commenced.

```
y = [+|-]n;
x = [+|-]n;
```

Alternative to specifying the position attribute. Uses the coordinates of the previous option in the menu description and performs a relative (if + or - is specified) or absolute move in the specified direction.

help = "*file*";

Attaches a help file to the menu option the file name can be a full path name of the name of a file in the standard help directory.

menu = "*menuname*";

Replaces the **action** attribute. It defines the name of a sub menu to be invoked. The sub menu must be defined in the same description file.

When a sub menu is selected only the screen area of the encompassed by that sub menu is updated. Therefore, visually cascading menus can be easily developed.

valid = "*expression*";

An expression which determines whether an option is valid. For example:

```
valid = (uid() == "andrew");
```

would allow the option to be selected only by the user with Unix login Id, andrew. Invalid options appear on the menu (in a different colour or highlight mode) but cannot be selected. Validation conditions are recalculated after an action is performed.

The standard operators used by the forms editor expression grammar are available. Refer to the KE Texpress Design Guide for a complete description.

Several in built functions may be used in the expression. They are:

`columns()`	Returns number of screen columns.
`exists(`*`filename`*`)`	Returns true if file filename exists.
`gid()`	Returns Unix group id of current user.
`lines()`	Returns number of screen lines.
`size(`*`filename`*`)`	Returns size of file filename.
`strlen(`*`str`*`)`	Returns length of string *str*.
`system(`*`prog`*`)`	Runs the program *prog* and returns its numeric exit status.
`uid()`	Returns Unix login id of current user.

visible = "*expression*";

Similar to the **valid** attribute except that if visible is not true, the option does not appear on the menu. Relative option positions are calculated from the last visible option. Visible condition are recalculated after an action is performed

exit;

Replaces the **action** attribute and causes the menu to exit when the option is selected. If the menu is a sub menu, the previous menu is resumed, otherwise the menu program terminates.

An example menu description with various option components is:

```
option
{
        position = (2, centre);
        title = "First database";
        selection = "fF^F";
        action = "te xforms first";
        message = "Commencing first database";
        help = "/usv/help/first option";
        valid = (uid() == "andrew" || uid() == "ian");
}
option
{
        y = +1;
        title = "Second database (query only)";
        selection = "sS";
```

```
        action = "texforms -y second";
        message = "Commencing second database";
        visible = (uid() == "andrew" &&
        exists("/tmp/afile"));
}
option
{
        y = +2;
        title = "Sub Menu";
        selection = "mM";
        menu = "sub";
}
option
{
        y = +2;
        title = "Exit";
        selection = "xX^X";
        exit;
}
```

The last two options would invoke a sub menu or exit the program respectively.

Escape Component

An escape component can define the **selection**, action and **message** attributes. Escape components are generally used to invoke a shell from within the menu.

An example escape component is:

```
escape
{
        selection = "!";
        action = "/bin/sh";
}
```

Menu Option Selection

Menu options are selected by moving the cursor to the appropriate option and then typing the↵ key.

↓ Move the cursor to the next menu option.

↑ Move the cursor to the previous menu option.

↵ Select the current option and execute its associated command.

The menu description file may define various command characters. If such command characters are typed, they result in the cursor jumping directly to the option associated with that character.

There is no exit command implicitly defined in the front end menu programs. Therefore the menu description file should contain an appropriate exit command definition.

Chapter 5

Label Printing on a PostScript Printer

Overview

KE Texpress provides printing utilities that take advantage of the PostScript capabilities of a printer, when presenting data from a KE Texpress database. The appearance of the printed data on the label may be specified in detail. It is possible to specify the font, format and point size of data objects, and to stipulate the relative or absolute positions of blocks of data, as well as data objects within a block.

The KE Texpress PostScript label program is called texlabel. Another utility called texbarcode may be used in conjunction with texlabel to produce barcode labels.

Printing Labels

The texlabel command is used in conjunction with a file containing a specification of the appearance of the data on the label, and a file containing a description of the layout of the labels on each page. The data to be printed may be supplied as standard input, or taken from a file of values. By default, the label printing begins on the top left label on the page, but may be made to begin on a particular label. The specification of the appearance of the data on the label generally uses the standard fonts known to the printer, but may include a customised font, supplied in a file.

The format of the command is:

```
texlabel -s spec -ppagedesc [-vvals] [-n labnum] [-f font]
```

When texlabel is used as an output command in report generation within KE Texpress, the values data is provided as standard input, so the -v*vals* option is not used.

The texlabel command produces as output a stream of PostScript. Labels may be printed by piping the texlabel command to the appropriate command for printing a PostScript file, for example:

```
texlabel -s spec -ppagedesc | lp -dps
```

The command line options are as follows:

`-sspec`

Use the file spec as the specification of the layout of the data on the labels. This file contains specifications of properties such as position, font, format and size for blocks of data and data objects within blocks.

`-p`*`pagedesc`*

Use the compiled version of the file *pagedesc* as the description of the layout of the labels on each page. This compiled version is in the directory ~texpress/etc/labels/pages. It is compiled from the description source file, using the texmessages command as described later in this Chapter.

`-v`*`vals`*

Take the values data from the file *vals* instead of from standard input.

`-f`*`font`*

Use the font specification in the file *font*. Provision of such a font specification is necessary only if the label specification file, *spec*, uses a font not available in the printer itself.

`-n`*`labnum`*

Begin the label printing on label number *labnum*. This number is determined counting from left to right on a row of labels, for each row in turn. The top left label is number 1.

Values Data

The data to appear on each label consists of any required combination of:

- Fixed strings, which do not vary from one label to another
- Strings containing the values of named data elements, which do vary from one label to another.

The specification contains the names of each data element to be printed, and the values data provides the value taken by this named data element for each label in turn, in a statement:

```
name=value
```

When the values data is produced by KE Texpress report generation, the values are the contents of fields in a KE Texpress database. Any unique name may be chosen for a data element. The KE Texpress item id is often a convenient name to choose.

Each line in the values data is a simple name=value statement, with the exception of each line indicating the end of the data for one label. This delimiting line must consist of a tilde character ("~") on a line by itself . Values data for producing address labels might contain the following:

```
title=Mr
fstname=Jack
surname=Robinson
comp=Clevercorp Pty Ltd
dept=Marketing Dept
addr1=Level 2
addr2=Clevercorp Building
addr3=10 Hill St
town=NEWTOWN
state=VIC
pcode=3999
~
```

A simple KE Texpress report form may be designed to report data in this format.

Page Description File

In the directory:

```
~texpress/etc/labels/pages
```

the KE Texpress software distribution provides two sample page description files, **ps24addr** and **ps6addr**. In a page description file, all blank lines and comment lines beginning with a # character are ignored. All other lines have the form:

```
nnn     "value" # explanatory comment
```

where nnn is the number used by KE Texpress to identify uniquely the value "**value**".

Installing a Page Description File

It is advisable not to alter the original distribution page description files. The user should copy one of the originals should be copied using a Unix command of the form:

```
cp ~texpress/etc/labels/pages/ps24addr   filename
```

The file called filename, which contains a copy of the original page description, can be edited using a standard Unix editor. The file ps24addr describes an A4 page of 24 labels. However, different brands of label pages have small differences in the layout of the labels on the page and it may be necessary to use a slightly different page description to suit the brand of label page to be used.

To install the new page description file, the user should give the command:

```
texmessages -d  filename
```

The compiled version of the page description in filename will be stored in the file filename.map in the pages directory.

Page Description Option

Someone wanting always to use the same page description file, filename, may find it convenient to specify the KE Texpress option:

```
pagedesc= filename
```

in either the **TEXPRESSOPTS** or the **dbnameopts** environment variables. Once this is done, there is no need to use the -ppagedesc argument to the

tlabcommand (see the KE Texpress Design Guide for more details on database options).

Label Specification File

The KE Texpress software distribution includes two sample label specification files in the directory:

```
~texpress/etc/labels/specs
```

The file labspec contains a sample label specification for producing address labels. The file barspec contains a sample labels specification for producing barcode labels (see Section 8.5.).

Label Specification Design

The basic specification has the format:

```
<default format list>
blocklist
```

where the blocklist is a sequence of one or more blocks, each of which has the format:

```
block=blockname
{<block format list>
        objectlist
}
```

The objectlist specifies the sequence of data objects in the block. Each object is a sequence of data elements as follows:

```
[object format list] element1 element2 ... elementn;
```

If the data element is a fixed string, it appears in the label specification between double quotation marks. If the data element's content varies from label to label according to the value in the values data, the name of the data element appears in the label specification, separated by a colon from the maximum number of characters to be allowed for the value. Examples of data elements are shown below

`"Name:"`	*A fixed string (constant for all labels)*
`name:20`	*A named data element of up to 20 characters*

The file labspec in the directory ~texpress/etc/labels/specs contains an example of a specification for an address label using the example values data given earlier in this Chapter. This example specification is shown below:

```
<font=Times format=Roman size=10>
block=address
{<lev=centre lat=centre kill=0>
        [depth=0] title:10 fstname:20 surname:20;
        [depth=+10] comp:10;
        [depth=+10] dept:10;
        [depth=+10] addr1:50;
        [depth=+10] addr2:50;
        [depth=+10] addr3:50;
        [depth=+10 join=" "] town:20 state:20 pcode:10;
}
```

Data Blocks

The data on the label should fall logically into subsets, each of which can be positioned on the label as a block of data. For example, there may be one block in the top left corner of the label, another centrally on the label, and yet another in the bottom right corner of the label.

In the case of an address label there is only one block of data, and in the specification above it is positioned centrally on the label. Each of the blocks must be named. It is possible to locate a block centrally between two other data blocks, which have to be referred to by name. Any name meaningful to the specification designer is sufficient, provided it is unique among the blocks in the label.

Data Objects

A data object may contain a single data element or a sequence of data elements which are to be located together, with their contents concatenated, and which may as a whole be described by a common format list. In the example specification above, the first object contains three data elements, as does the last object, but all other objects contain a single data element.

Format Lists

A format list is a sequence of white space separated statements of the form:

```
property=value
```

Default Format

Any format settings that apply to most or all of the label may be listed in the default format list at the beginning of the label specification. In the example

specification, since the font, format and point size are common to the entire label, these properties appear in the default format list.

Block Format

Any format settings that involve changes to settings listed in the default format list and apply to most or all of the data in a block, may be listed in the format list, for that block. The properties referring specifically to block location and block composition may appear appropriately in a block format list. If they hold for most or all of the blocks on the label, they may appear in a default format list. They should not appear in an object format list as they have no meaning there. These properties are:

`lev`	The vertical position of the block on the label
`lat`	The horizontal position of the block on the label.
`kill`	The list of numbers of any objects which when empty should be regarded as non existent in the object list.

Object Format

Format settings that involve changes to settings listed in the block format list and apply to the data in an object should be listed in the format list for that object The properties referring specifically to object location within a block may appear appropriately in an object format list. If they hold for most or all of the objects in a block, they may appear in a block format list. These properties are:

`depth`	The vertical position of the object in the block.
`pos`	The horizontal position of the object in the block.

Format Properties

The available format properties and the possible settings to be used with them are listed below:

font Specifies the font to be used. Examples of usage are:

```
font=Times
font=Courier
```

`format` Specifies the format within a font. There are five possible values:

`format=Roman`	If the font is Times.
`format=Plain`	If the font is not Times.
`format=Italic`	
`format=Bold`	

```
format=BoldItalic
```

`size` Specifies the point size of the type required. An example of the usage is:

```
size=10
```

`lev` Specifies the level, or vertical position, of a block on the label. There are several absolute settings and some relative ones. Allowable forms of usage are:

`lev=top`
: At the top of the label.

`lev=bottom`
: At the bottom of the label.

`lev=centre`
: Centred vertically on the label.

`lev=centre(a,b)`
: Centred vertically between blocks a and b.

`lev=centre(a,)`
: Centred vertically between block a and the bottom margin on the label.

`lev=centre(,b)`
: Centred vertically between the top margin of the label and block b.

`lev=at:10`
: At a position 10 points below the top margin on the label.

`lev=at:+10`
: At a position 10 points below the top of the previous block.

`lev=sep:10`
: At a position 10 points below the bottom of the previous block, i.e. so that there is a vertical separation of 10 points between the blocks.

`lat` specifies the lateral, or horizontal, position of a block on the label. There are several absolute settings and some relative ones. Allowable forms of usage are:

`lat=left`
: At the left margin of the label.

`lat=right`
: At the right margin of the label.

`lat=centre`
: Centred horizontally on the label.

`lat=centre(a,b)`
: **C**entred horizontally between blocks a and b.

`lat=centre(a,)`
: Centred horizontally between block a and the right hand margin on the label.

`lat=centre(,b)`
: Centred horizontally between the left hand margin of the label and block b.

`lat=at:10`
: At a position 10 points to the right of the left margin on the label.

`lat=at:+10`
: At a position 10 points to the right of the left extremity of the previous block.

`lat=sep:10`
: At a position 10 points to the right of the right extremity of the previous block, i.e. so there is a horizontal separation of 10 points between the blocks.

`depth` Specifies the vertical position of an object within a block. Its value, given in points, may be absolute, i.e. measured from the top of the block, or relative, i.e. measured from the position of the previous object in the block. Examples of usage are:

`depth=10`
: At 10 points below the top of the block

`depth=+10`
: At 10 points below the previous object in the block.

`pos` Specifies the horizontal position of an object within a block. An object may be at left or right extremity of the block, or centred within the block. Alternatively it may be begin at a specified horizontal indent in the block, or be indented relative to the start of the previous object, or be separated by a specified distance from the end of the previous object. Its value, given in points, may be absolute or relative. Allowable forms of usage are:

`pos=ljust`
: Left justified within the block

`pos=rjust`
: Right justified within the block

`pos=cjust`
: Centred horizontally within the block

`pos=indent:10`

Starting 10 points to the right of the left extremity of the block.

`pos=indent:+10`

Starting 10 points to the right of the start of the previous object in the block.

`pos=sep:10`

Starting 10 points to the right of the end of the previous object, i.e. at a separation of 10 points from the previous object.

`kill` Specifies that certain objects are to be omitted from the list of objects when they contain no data. The kill property is intended to be used in conjunction with relative values for depth, to avoid the appearance of blank lines in the printed output. In the address example, if there is no company name for one label, and the object consisting of the named element comp is killed, then the object list for that label will no longer contain that object. The following object (addr1) will be output 10 points lower than the object now preceding it in the object list, (title, fstname, surname), avoiding a blank line in the printed address.

The value taken by kill may be just a single number, or a list of numbers separated by commas. Examples of usage are:

`kill=4`

Kill object 4 in the block if empty.

`kill=1,3,5`

Kill objects 1, 3, or 5 if empty.

`kill=0`

Kill any object if it is empty.

`erase` Specifies that certain data elements are not to be printed if certain other objects contain no data. It is used typically when a fixed string acting as a label for some part of the data is not required in the printed output when that data to which it refers is not present.

The usage of the erase property varies according to whether it is used in an object format or a block format.

It is used in an object format when both the elements tested for emptiness and those erased because of it are all in the same object. The value of the erase property then consists of two colon separated lists of comma separated numbers. Examples of usage in an object format are:

`erase=3:2`

If element 3 is empty then do not print element 2 either.

`erase=2,3:1,4`

If elements 2 and 3 are empty, then do not print elements 1 and 4 either.

It is used in an block format when the elements tested for emptiness and those erased because of it are not all in the same object. The value of the erase property then consists of two colon separated lists of comma separated number pairs of the form *mm*/*nn*. Such a pair refers to element *nn* in object *mm*. Examples of usage in a block format are:

`erase=2/1:1/1`

If element 1 in object 2 is empty then do not print element 1 in object 1 either.

`erase=2/1,2/2:1/1,1/4`

If elements 1 and 2 in object 2 are empty, then do not print elements 1 and 4 in object 1 either.

`join` Specifies that a particular string is to used between successive data elements. The default joining string is a single space, which is generally appropriate, but the join property allows redefinition of this string where required. For example in address labels, the join string between town and state, and between state and postcode could be made a double space. Examples of usage are shown below:

`join="  "`

Join data elements with a double space.

`join="----"`

Join data elements with four hyphens.

`wrap` Specifies the manner in which a string is to be wrapped if it is too large to fit horizontally on the label, or a limited part of the label. In using the wrap property, it is necessary to indicate three numeric values, the measurement in points of:

(i) The location at which wrapping is to occur
(ii) The location to which the wrapping is to occur
(iii) The relative depth at which the wrapped line is to continue.

The first two of these values may be either absolute, i.e. measured from the left extremity of the block, or relative to the start of the object being wrapped. The last value is relative by definition. It specifies the depth in points of the continuation line relative to that prior to the wrap. Examples of usage are:

`wrap=40,10,10`

Wrap at 40 points to the right of left extremity of the block. Begin the wrapped portion 10 points to the right of the left extremity of the block and at 10 points lower than the line depth before the wrap

`wrap=+40,10,10`

Wrap at 40 points to the right of the start of the object. Begin the wrapped portion 10 points to the right of the left extremity of the block and at 10 points lower than the line depth before the wrap.

`wrap=40,+10,10`

Wrap at 40 points to the right of the left extremity of the block. Begin the wrapped portion 10 points to the right of the start of the object and at 10 points lower than the line depth before the wrap.

`wrap=+40,+10,10`

Wrap at 40 points to the right of the start of the object. Begin the wrapped portion 10 points to the right of the start of the object and at 10 points lower than the line depth before the wrap.

Either of the last two examples show typical usage.

`phrase` Specifies the start and end of a succession of data objects that are to be regarded as a phrase not to be broken during a wrap. The property is used only in conjunction with the wrap property. The phrase property settings may be used only in object formats and only in pairs, though both parts of the pair need not be in the same object format. The two allowable settings are used as follows:

`phrase=start`

The phrase is to start at the beginning of the object.

`phrase=stop`

The phrase is to stop at the end of the object.

Barcode Labels

Barcode Values Data

The **texbarcode** utility may be used to create values data which can then be used to produce barcode labels. A file of values data, barvals, may be created by giving the command:

```
texbarcode > barvals
```

The utility prompts the user as follows:

```
Enter header string:
```

The user may enter any data to be printed above the barcode. For example, if the barcode is to contain a Registration Number, the required header string might be Registration No. If no explanation is required above the barcode, no string need be entered, return is typed immediately.

The user is then prompted as follows:

```
Enter barcode string:
```

The characters in the barcode are now entered, e.g. 200416.

There are repeated prompts for first header string, then barcode string, until the user terminates the input by typing Ctrl+D as the first character of an input line.

The barvals file then contains values data of the form:

```
desc=Registration No.
bcode=200416
transl="200416"
~
```

Barcode Label Specification

The KE Texpress software distribution includes, in the directory:

```
~texpress/etc/labels/specs
```

the file barspec, containing a sample barcode label specification. The content of the file is as follows:

```
<lat=centre lev=centre>
block=barcode
{<pos=cjust>
        [depth=0 font=Times format=Roman size=15]
        desc:30;
        [depth=+40 font=Ba rcode39 format=Plain size=30]
                                            bcode:30;
        [depth=+20 font=Times format=Roman size=10]
        transl:30;
}
```

This specification is suitable for use with the values data created by the texbarcode utility.

Creating Barcode Labels

In the barspec specification file, a font, Barcode39, is specified. This is not a font supplied with a laser printer. It is a PostScript definition of a font supplied with the KE Texpress software distribution, in the file Barcode39 in the directory ~texpress/etc/labels/fonts. When this special font is to be used, the -f argument to the texlabel command is required. To produce barcode labels using the barspec specification and the barvals data, the required command is:

```
texlabel -s~texpress/etc/labels/specs/barspec -pps24addr
-fBarcode39 -vbarvals | appropriate print command
```

It is not necessary to create the barvals file. A suitable pipeline is:

```
texbarcode | tlab -s~texpress/etc/labels/specs/barspec -
pps24addr -fBarcode39 | appropriate   print command
```

Index

A

B

C

D

E

F

H

I

L

M

O

P

R

S

T

U