Using the Desktop Database

You can manipulate the desktop database with a set of low-level routines that follow the parameter-block conventions used by the **File Manager**.

The desktop database functions use the **DTPBRec** parameter block.

Because you cannot use the **Desktop Manager** functions on a disk that does not have a desktop database, call **PBHGetVolParms** to verify that the target disk has a desktop database before calling any of the **Desktop Manager** functions.

Because the **Finder** uses the desktop database, the database is almost always open. When the **Desktop Manager** opens the database, it assigns the database a reference number that represents the access path. Use the **PBDTGetPath** function to get the reference number, which you must specify when calling most other **Desktop Manager** functions (see Desktop Manager Routines). If the desktop database is not open, **PBDTGetPath** opens it.

If you are manipulating the database in the absence of the **Finder**, you can open the database with **PBDTOpenInform**, which performs the same functions as **PBDTGetPath** and also sets a flag to tell your application whether the desktop database was empty when it was opened. Your application should never close the database.

The **Desktop Manager** provides different functions for manipulating different kinds of information in the database. Not all manipulations are possible with all kinds of data.

You can retrieve five kinds of information from the database:

- icon definitions
- file types and icon types supported by a known creator
- name and location of applications with a known creator
- user comments for a file or a directory
- size and parent directory of the desktop database

To retrieve an icon definition, call **PBDTGetIcon**. You must specify a file creator, file type, and icon type. The database recognizes both large and small icons, with 1, 4, or 8 bits of color encoding. (See the description of icons in **Finder-Related Resources** for details.)

To step through a list of all the icon types supported by an application, make repeated calls to **PBDTGetIconInfo**. Each time you call **PBDTGetIconInfo**, you specify a creator and an index value. Set the index to 1 on the first call, and increment it on each subsequent call until **PBDTGetIconInfo** returns the result code afpItemNotFound. For each entry in the icon list, **PBDTGetIconInfo** reports the icon type, the file type it is associated with, and the size of its icon data.

To identify the application that can open a file with a given creator, call **PBDTGetAPPL**. In each call to **PBDTGetAPPL**, you specify a creator (which is the application's signature) and an index value. An index value of 0 retrieves the “first choice” application-that is, the one with the most recent creation date. By setting the index to 1 on the first call and incrementing it on each subsequent call until **PBDTGetAPPL** returns the result code afpItemNotFound, you can make multiple calls to **PBDTGetAPPL** to find all copies or versions of the application with this signature on the disk. **PBDTGetAPPL** returns them all in arbitrary order. **PBDTGetAPPL** returns the name, parent directory ID, and creation date of each application in the desktop database.

To retrieve the user comments for a file or directory, call **PBDTGetComment**. The user can change comments at any time by typing in the comment box of the information window for any desktop object.
Your application should not ordinarily call the functions for adding and removing data to and from the database. If your application does need to write to or delete information from the desktop database, it must call **PBDTFlush** to update the copy stored on the volume.

The following list summarizes the data manipulation functions.

Icon definitions

read:	**PBDTGetIcon**
write:	**PBDTAddIcon**
remove:	–

Icon types supported by an application

read:	**PBDTGetIconInfo**
write:	–
remove:	–

Applications with a given creator

read:	**PBDTGetAPPL**
write:	**PBDTAddAPPL**
remove:	**PBDTRemoveAPPL**

User comments

read:	**PBDTGetComment**
write:	**PBDTSetComment**
remove:	**PBDTRemoveComment**

Entire desktop database

read:	**PBDTGetInfo** (size and parent of the database)
write:	**PBDTFlush** (updates the copy stored on the volume)
remove:	**PBDTDelete** and **PBDTReset** (not called by you)