B. Application APIs

You can extend some Be applications by providing them with add-on modules, which they will load and integrate into the set of features they provide for the user. Currently, the Browser is the only Be application with a public API for add-on extensions.

The Browser

The Browser can accept add-on modules that deal with database records and that can be invoked from menu items to carry out discrete tasks. The modules should be compiled as on-add images (described in *The Kernel Kit* chapter), and should be placed in the **/system/add-ons/Browser** directory.

The Browser will create an item for its Add-Ons menu with the same name as the add-on file. If the name ends in a hyphen plus one character, that character will be the keyboard shortcut for the item. For example, if the file is

/system/add-ons/Browser/Recede in Time-r

the Browser will add a "Recede in Time" item to its Add-Ons menu and assign it Command-r as a keyboard shortcut. The shortcut should not conflict with any that the Browser already uses.

The Browser loads the add-on module whenever the user operates the item. The add-on must provide the Browser with a single entry point, a function named **process_refs()**. It has the following syntax:

void **process_refs(**record_ref *directory*, BMessage **message*, void **data***)**

The *directory* is a reference to the directory the Browser is currently displaying in the active window. The *message* is a standard **B_REFS_RECEIVED** BMessage. It has a "refs" entry with **record_ref**s for all the items in the directory that are currently selected. The *data* argument is unused at present; ignore it.

After it loads the add-on image, the Browser creates a thread for it and calls its **process_refs()** function in that thread. When **process_refs()** returns, the Browser unloads the image. The add-on should make sure that any additional threads that it spawned are destroyed before it returns—especially any windows it displayed to the user.

< The Browser invokes **process_refs()** each time the user operates the menu item. If the user operates the item a second time before the first invocation of **process_refs()** returns, two instances of the function will be executing, each in its own thread. Unfortunately, when either instance returns, the Browser will unload the add-on image, leading to predictable undesired consequences. This is a known bug that will be repaired in a future release. >

To compile the add-on image, follow the directions for shared libraries in the Metrowerks *CodeWarrior* manual. In summary, you should specify the following options to the linker:

- **–G**, to tell the linker to produce an add-on image.
- **–export pragma**, to tell it that your source code has **#pragma** directives exporting the **process_refs()** symbol. Then surround the definition of the function with directives that turn exporting on and off:

```
#pragma export on
void process_refs(record_ref dir, BMessage *msg, void *data)
{
    . . .
}
#pragma export off
```

  This gives the Browser the access it needs to call the function.

You can link the module against the system library; you shouldn't link it against the Browser.

Once compiled, place the module in the **/system/add-ons/Browser** directory, as discussed above. This is the only place the Browser looks for modules to load.