Creating a Subscriber

You need to create a **Subscribe To** menu command in the **Edit** menu. When a user chooses **Subscribe To** from this menu, your application should display the subscriber dialog box on the user's screen.

Use the **NewSubscriberDialog** function to display the subscriber dialog box on the user's screen. This function is similar to the **CustomGetFile** procedure described in the **Standard File Package** to get information from the user, such as the name of the edition being subscribed to. The dialog box displays a listing of all available editions and allows the user to see a preview (thumbnail sketch) of the edition selected.

The subscriber dialog box allows the user to choose an edition to subscribe to. The **NewSubscriberDialog** function handles all user interaction until a user clicks **Subscribe** or **Cancel**. When a user selects an edition container, the **Edition Manager** accesses the preview for the edition container (if it is available) and displays it.

You pass a NewSubscriberReply record as a parameter to the **NewSubscriberDialog** function.

The canceled field of the NewSubscriberReply structure returns a Boolean value of TRUE if the user clicked Cancel. To indicate which edition format types (text, graphics, or sound) your application can read, you set the formatsMask field to one or more of these constants:

kPICTformatMask	Can subscribe to 'PICT',
kTEXTformatMask	'TEXT', and
ksndFormatMask	'snd '.

To support a combination of formats, add the constants together. For example, a formatsMask of 3 displays both graphics and text edition format types in the subscriber dialog box.

The container field is of data type **EditionContainerSpec**. You must initialize the container field with the default edition volume reference number, directory ID, filename, and part. To do so, use the **GetLastEditionContainerUsed** function to obtain the name of the last edition displayed in the dialog box.

err = **GetLastEditionContainerUsed** (*container*);

This function returns the last edition container for which a new subscriber was created using the **NewSection** function. If there is no last edition, or if the edition was deleted, **GetLastEditionContainerUsed** still returns the correct volume reference number and directory ID to use, but leaves the filename blank and returns the fnfErr result code.

The container field is of structure type EditionContainerSpec.

The field theFile of the EditionContainerSpec structure is of type FSSpec. See the **File Manager** description for further information on file system specification records.

After filling in the fields of the new subscriber reply record, pass it as a parameter to the **NewSubscriberDialog** function, which displays the

subscriber dialog box.

err = **NewSubscriberDialog** (*reply*);

After displaying the subscriber dialog box, call the **NewSection** function to create the section record and the alias record.

If the subscriber is set up to receive new editions automatically (not manually), the **Edition Manager** sends your application a Section Read event. Whenever your application receives a Section Read event, it should read the contents of the edition into the subscriber.

The Listing below illustrates how to create a subscriber. As described earlier, you must set up and display the subscriber dialog box to allow the user to subscribe to all available editions. After your application creates a subscriber, your application receives a Section Read event to read in the data being subscribed to. Be sure to add the newly created section to your list of sections for this file. There are many different techniques for creating subscribers and unique IDs; this listing displays one technique.

```
//Listing. Creating a subscriber
// Assuming inclusion of MacHeaders
#include <Editions.h>
#include <AppleEvents.h>

// This is a sample declaration for the pointer to your document information.
typedef struct {
     short resForkRefNum;
     FSSpec fileSpec;
     SectionHandle  sectionH;
     short nextSectionID;
     } *MyDocumentInfoPtr;

// Prototype your function like this prior to calling it
void DoNewSubscriber(MyDocumentInfoPtr);


void  DoNewSubscriber(MyDocumentInfoPtr thisDocument)


{
   OSErr getLastErr;
   OSErr dialogErr;
   OSErr sectionErr;
   short resID;
   SectionHandle thisSectionH;
   NewSubscriberReply    reply;

   // User defined prototypes
   void MyErrHandler(OSErr);
   void AddSectionAliasPair(MyDocumentInfoPtr,SectionHandle, short);


    // Put default edition name into reply record.
```

```
    getLastErr = GetLastEditionContainerUsed(&reply.container);

    // Can subscribe to pictures or text.
    reply.formatsMask = kPICTformatMask + kTEXTformatMask;

    // Display dialog box and let user select}
    dialogErr = NewSubscriberDialog(&reply);

    // There's usually no error returned here, but if there is,
    // then it makes no sense to continue with this operation.
    // Pass control to MyErrHandler.
    if ( dialogErr )
        MyErrHandler(dialogErr);

    // Do nothing if user canceled.
    if ( reply.canceled )
        return;

    // Advance counter to make a new unique sectionID for this
    // document. It is not necessary to equate section IDs with
    // resources.
    thisDocument->nextSectionID++;

    // Create a subscriber section.
    sectionErr = NewSection(&reply.container,
                                        &thisDocument->fileSpec,
                                        stSubscriber,
                                        thisDocument->nextSectionID,
                                        sumAutomatic, &thisSectionH);
    if ( sectionErr )
        // Same reasoning as above. If a new section could not be
        // created, don't continue with this operation. Pass
        // control to MyErrHandler.
        MyErrHandler(sectionErr);

    resID = thisDocument->nextSectionID;

    // Add this section/alias pair to your internal bookkeeping.
    // AddSectionAliasPair is a routine to accomplish this.
    AddSectionAliasPair(thisDocument, thisSectionH, resID);

    // Remember that you will receive a Section Read event to read
    // in the edition that you just subscribed to because the initial
    // mode is set to sumAutomatic.

    // Remember that the section and alias records need to be saved
    // as resources when the user saves the document.

}
```