Opening and Closing a Document Containing Sections

When opening a document that contains sections, your application should use the **GetResource** function to get the section record and the alias record for each publisher and subscriber. Set the alias field of the section record to be the handle to the alias.

You also need to register each section using the **RegisterSection** function. The **RegisterSection** function informs the **Edition Manager** that a section exists.

err = **RegisterSection (***sectionDocument, sectionH, aliasWasUpdated***);**

The **RegisterSection** function adds the section record to the **Edition Manager**'s list of registered sections. This function assumes that the alias field of each section record is a handle to the alias record. The alias record is a reference to the edition container from the section's document. If the **RegisterSection** function successfully locates the edition container for a particular section, the section is registered through a shared control block. The control block is a private field in the section record.

If the **RegisterSection** function cannot find the edition container for a particular subscriber, **RegisterSection** returns the containerNotFoundWrn result code. If the **RegisterSection** function cannot find the edition container for a particular publisher, **RegisterSection** creates an empty edition container for the publisher in the last place the edition was located. The **Edition Manager** sends your application a Section Write event for that section.

When a user attempts to open a document that contains multiple publishers to the same edition, you should warn the user by displaying an alert box.

When a user opens a document that contains a subscriber (with an update mode set to automatic), receives a new edition, and then closes the document without making any changes to the file, you should update the document and simply allow the user to close it. You do not need to prompt the user to save changes to the file.

When closing a document that contains sections, you must unregister each section (using the **UnRegisterSection** function) and dispose of each corresponding section record and alias record.

err = **UnRegisterSection (***sectionH***);**

The **UnRegisterSection** function removes the section record from the list of registered sections and unlinks itself from the shared control block.

The following Listing illustrates how to open an existing file that contains sections. As described earlier, you should retrieve the section and alias resources, connect the pair through the alias field of the section record, and register the section with the **Edition Manager**. There are many different techniques for retrieving resources; this listing shows one technique. If an alias was out of date and was updated by the **Alias Manager** during the resolve, the **Edition Manager** sets the *aliasWasUpdated* parameter of the **RegisterSection** function to TRUE. This means that you should save the document. Additionally, your application must maintain its own list of registered sections for each open document that contains sections.

```
// Listing. Opening a document containing sections
// Assuming inclusion of <MacHeaders>

#include <Editions.h>

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


// This is how you would prototype this routine.
void OpenExistingDocument(MyDocumentInfoPtr);

void OpenExistingDocument(MyDocumentInfoPtr thisDocument)

{

    SectionHandle  sectionH;  // Handle to section
    AliasHandle aliasH;      // Handle to alias in section handle
    Boolean     aliasWasUpdated; // alias updated or not?
    OSErr registerErr;       // typical error checking variable
    short resID;             // resid of rSectionType
    short thisone;           // index into sections
    short numberOfSections;  // number of sections to index
    Str255 aName;            // name of resource
    ResType rType = rSectionType;   // resource type for section

    // User defined function prototypes.
    void MyAddSectionAliasPair(MyDocumentInfoPtr ,SectionHandle,short);
    void AliasHasChanged (SectionHandle);

    // Set the curResFile to be the resource fork of thisDocument.
    UseResFile(thisDocument->resForkRefNum);

    // Find out the number of section resources.
    numberOfSections = Count1Resources(rSectionType);

    // In determining the number of section/alias resource pairs to
    // get, this code only loops for as many sections it finds.
    // It is unusual to have more section resources than alias
    // resources. Your code may want to check this and handle it
    // appropriately. You now have a count of the number of section/alias
    // resource pairs to get. Loop to get them, connect them, and register
    // the section.

    for (thisone = 1; thisone <= numberOfSections; thisone++) {


        sectionH = (SectionHandle)Get1IndResource(rSectionType,thisone);
        // If sectionH is NIL, something could be wrong with the file.
```

```
            // Be sure to check for this at this point.

            // Get the resource ID of the section and use this to get the
            //  alias with the same resource ID.
            GetResInfo((Handle)sectionH, &resID, &rType, aName);

            //Detaching is not necessary but it is convenient.
            DetachResource((Handle)sectionH);

            aliasH = (AliasHandle)Get1Resource(rAliasType, resID);
            // If aliasH is NIL, then there could be something wrong
            // with the file. Be sure to check for this at this point.

            // Detaching is not necessary, but it is convenient.
            DetachResource((Handle)aliasH);

            // Connect section and alias together.
            (*sectionH)->alias = aliasH;

            // Register the section.
            registerErr = RegisterSection(&thisDocument->fileSpec,sectionH,
                        &aliasWasUpdated);

            // The RegisterSection function may return an error if a section
            // is not registered. This is not a fatal error. Continue looping
            // to register remaining sections.

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

            // If the alias has changed, make note of this. It is
            // important to know this when you save. AliasHasChanged is a
            // routine that will do this.
            if (aliasWasUpdated)
                AliasHasChanged(sectionH);
        }
    }
```