TECHNOTE: On Launching an App with a Document

By John Montbriand

Internet address: **tinyjohn@sasknet.sk.ca**
Apple Developer Technical Support (DTS)

This Note addresses the issue of how to launch another application from inside your app and then open a document using that app. Preferably, developers would like to utilize the signature information from the document to find the application.

Developers writing applications that sub-launch other applications will find the material contained herein lets them easily extend the functionality of their applications to include the ability to launch applications with the specific documents they want to be opened.

This Note combines information found in the chapter "Creating and Sending Apple Events" in *Inside Macintosh: Interapplication Communications* and the chapter "Desktop Manager" in *Inside Macintosh: More Macintosh Toolbox*. The format of the Open Documents event is described in the section "Handling the Required Apple Events" (4-11) in *Inside Macintosh: Interapplication Communication.*

About Launching an Application with a Document

To launch an application with a document, the following steps are required:

1. Find the appropriate application for opening the document
2. Try sending the open documents event to the running application, and
3. If Step 2 fails, launch the application with the appropriate parameters.

Finding the Right Application for the Document

Every document has type and creator bytes associated with it. The first step, therefore, in finding what application to use is to take a look at the creator bytes. You can find this value by using the `FSpGetFInfo` call:

```
err = FSpGetFInfo(&spec, &fndrInfo);
```

`fndrInfo.fdCreator` contains the creator bytes for the document.

When specifying the destination address for the open document Apple event, the creator bytes are used to create a descriptor containing a `typeApplSignature` record. If the `AESend` indicates that no such application with that creator is running, then the creator bytes can be used to search the desktop files on attached volumes for the correct application to launch.

Using a Preferred Application to Open the Document

Sometimes, you may want to open a document using an application other than the one referred to by its creator bytes. For example, you may want to open every text document you encounter using a program such as ClarisWorks. In that case, you need to provide some internal means of mapping file types to creator types inside of your application that overrides the creator bytes returned by `FSpGetFInfo`. The code snippet in this Technote provides a means for doing this by allowing you to define a mapping function for re-mapping the creator type used to open a document.

Here is how you would define a simple mapping function that will work with `LaunchTheDocument`:

```
void MyMapCreator(FSSpec *document, OSType type, OSType *creator) {
    if (type == 'TEXT')
        *creator = 'BOBO'; /* open all text files with Claris Works,
                              creator = 'BOBO' */
}
```

Mapping functions receive three parameters: a FSSpec record referring to the document itself — in case you want to look at the document's data or name to establish its type — and the document's type and creator bytes. Mapping functions provide a simple way for your application to change the creator bytes that will be used to find the appropriate application before the search for the application begins. In this example, all documents of type `'TEXT'` are re-mapped to the creator bytes for ClarisWorks, telling `LaunchTheDocument` to ask ClarisWorks to open all text documents. The collected fragments in a code snippet provided later in this Technote give you an example of how to define a creator mapping function.

Sending the Open Documents Event

As described in *Inside Macintosh: Interapplication Communication* (4-13), the open documents event contains one list of alias records referring to documents to be opened by the target application. The following description takes you through the step-by-step construction of such an event.

First, create an address descriptor record (target_desc), using the creator bytes (`fndrInfo.fdCreator`):

```
    err = AECreateDesc(typeApplSignature, (Ptr) &fndrInfo.fdCreator,
                       sizeof(OSType), &target_desc);
```

Then you use this address record to specify the target application when you create the Apple event (the_apple_event):

```
    err = AECreateAppleEvent(kCoreEventClass, kAEOpenDocuments,
        &target_desc, kAutoGenerateReturnID, kAnyTransactionID,
        &the_apple_event);
```

Because you are opening a single document, you need to create a list containing an alias record referring to the file you want to open. Even though you're only asking the target application to open one document, you still create

a list containing only one alias record to conform to the definition of an open documents Apple event.

First, you create a list (files_list):

```
err = AECreateList(NULL, 0, false, &files_list);
```

and a descriptor record (`file_desc`) containing an alias to our document (the_file):

```
err = NewAlias(NULL, &the_file, &the_alias);

HLock((Handle) the_alias);
err = AECreateDesc(typeAlias, (Ptr) (*the_alias),
                    GetHandleSize((Handle) the_alias), &file_desc);
HUnlock((Handle) the_alias);
```

Then, you add the descriptor record (`file_desc`) containing the alias to the list (files_list):

```
err = AEPutDesc(&files_list, 0, &file_desc);
```

Now that you have the Apple event (the_apple_event), and the list (files_list), containing an alias record referring to the file you want opened, you can take the list and put it into the Apple event (the_apple_event) as an Apple event parameter:

```
err = AEPutParamDesc(&the_apple_event, keyDirectObject, &files_list);
```

At this point the open documents event (the_apple_event) has been set up correctly and you can send it to the application:

```
err = AESend(&the_apple_event, &the_reply, kAENoReply,
        kAENormalPriority,  kNoTimeOut, NULL, NULL);
```

When the application is not running

When `AESend` returns an error result of connectionInvalid, this means the application with the specified creator bytes is not currently registered with the Process Manager. In this case, the correct application must be launched before it can be sent the open document event. The first thing to do here is search

through the desktop files on connected volumes using the `PBDTGetAPPL` routine to find the application with the correct creator bytes for the document.

The following code fragment illustrates how this can be done.

```
    HVolumeParam vol_pb;
    DTPBRec desktop_pb;
    FSSpec application;
    vol_pb.ioNamePtr = NULL;
    err = fnfErr; /* default return value */
    vol_pb.ioNamePtr = NULL;
        /* for... every connected volume, 1, 2, ... */
    for (vol_pb.ioVolIndex = 1;
        PBHGetVInfoSync((HParmBlkPtr) &vol_pb) == noErr;
        vol_pb.ioVolIndex++) {
        /* PBHGetVolParms call to ensure the volume is a local volume */
            param_pb.ioVRefNum = vol_pb.ioVRefNum;
            if (PBHGetVolParmsSync((HParmBlkPtr)&param_pb) == noErr
            && volinfo.vMServerAdr == 0) {
            /* call PBDTGetPath to find the desktop file on the volume */
        desktop_pb.ioCompletion = NULL;
        desktop_pb.ioVRefNum = vol_pb.ioVRefNum;
        desktop_pb.ioNamePtr = NULL;
        desktop_pb.ioIndex = 0;
        if (PBDTGetPath(&desktop_pb) == noErr) {
            desktop_pb.ioFileCreator = app_creator_bytes;
            desktop_pb.ioNamePtr = application.name;
            if (PBDTGetAPPLSync(&desktop_pb) == noErr) {
                application.vRefNum = vol_pb.ioVRefNum;
                application.parID = desktop_pb.ioAPPLParID;
                /* at this point we have found the correct application */
                err = noErr;
                break;
            }
        }
    }
}
```

Note
This is a fairly simple search method. For your purposes, you may want to modify it so it does special things, such as not bothering with ejected disks that are still mounted, or searching particular disks before others. The code fragment searches local volumes only checking to see if they are local using the PBHGetVolParms routine. In the code snippet in this Technote, local volumes are searched before an optional search of non-local volumes takes place. ◆

Launching the App

Once you find the correct application, the next step is to coerce the Apple event that `AESend` failed to send into the correct format so a pointer to the parameters can be stored in the launch parameter block. This can be achieved with a call to `AECoerceDesc` specifying `typeAppParameters` as the destination type:

```
err = AECoerceDesc(&the_apple_event, typeAppParameters, &parameter_desc);
```

Once this coercion has been performed, everything required for setting up the launch parameter block is available so the application can be launched:

```
launch_pb.launchBlockID = extendedBlock;
launch_pb.launchEPBLength = extendedBlockLen;
launch_pb.launchFileFlags = 0;
launch_pb.launchControlFlags =
    launchContinue + launchNoFileFlags;
launch_pb.launchAppSpec = &application;
    /* launchAppParameters expects a pointer,
    so we lock and dereference the dataHandle
    stored in the parameter_desc */
HLock((Handle) parameter_desc.dataHandle);
launch_pb.launchAppParameters =
    (AppParametersPtr) (*parameter_desc.dataHandle);
err = LaunchApplication(&launch_pb);
HUnlock((Handle) parameter_desc.dataHandle);
```

If the Target Application is Not Apple Event Aware

If the target application is not high level event aware (pre-System 7), the Apple Event Manager will automatically coerce the event into the old style launch parameters when the program is being launched. The old style `'mstr'` and `'mst#'` resources used by MultiFinder in System 6 are no longer supported; therefore, you can't use these routines to open the document if the application is already running.

Collected Fragments in a Code Snippet

The following code snippet combines all of the concepts previously explained.

Note
The author has compiled and run the following with the MPW C, MPW PPC, SymantecC, MrC, and CodeWarrior C compilers and is satisfied that it is bug free for System 7.0 and onward. ◆

```
/* File SendOpen.c

Code snippet illustrating how to launch an application and send it
an open document Apple event.
by John Montbriand, 1995.
Copyright (C) 1995 by Apple Computer, Inc.  All Rights Reserved.*/

#include <Types.h>
#include <QuickDraw.h>
#include <Fonts.h>
#include <Windows.h>
#include <TextEdit.h>
#include <Dialogs.h>
#include <Menus.h>
#include <AppleEvents.h>
#include <StandardFile.h>
#include <Files.h>
#include <Errors.h>
#include <Aliases.h>
```

```
/* MapCreatorProcs, when provided as a parameter to LaunchTheDocument,
will be called to allow your app to re-map the creator type for a file
to a different creator type. Your function should either modify *creator
to refer to a preferred application by replacing its value with the
value of the preferred application's creator bytes, or leave *creator
untouched if you want the creator bytes from the document to be used. */

typedef void (*MapCreatorProc)(FSSpec *document, OSType type, OSType
                               *creator);

/* LaunchTheDocument attempts to open the document with either an
application that is already loaded in memory and running, or by finding
and launching the appropriate application with the document as a
parameter. If no application can be found with the creator bytes found
in the document, fnfErr is returned. Local Volumes are searched before
remote server volumes. Remote server volumes are only searched if
use_remote_apps is true. If the optional creator remapping procedure is
provided (remap != NULL), then it will be called to allow the caller to
re-map the creator bytes to a preferred application for the document's
type.. */

OSErr LaunchTheDocument(FSSpec *document, Boolean use_remote_apps,
                        MapCreatorProc remap) {
    OSErr err;
    AEAddressDesc target_desc; // address desc for target application
    AEDescList files_list;
    AEDesc file_desc, parameter_desc;
    AppleEvent the_apple_event;
    AppleEvent the_reply;
    AliasHandle the_alias;
    FInfo fndrInfo;
    OSType app_creator_bytes;
    LaunchParamBlockRec launch_pb;

/* initialize our records to NULL descriptors */
    AECreateDesc(typeNull, NULL, 0, &target_desc);
    AECreateDesc(typeNull, NULL, 0, &files_list);
    AECreateDesc(typeNull, NULL, 0, &file_desc);
    AECreateDesc(typeNull, NULL, 0, &the_apple_event);
    AECreateDesc(typeNull, NULL, 0, &the_reply);
    AECreateDesc(typeNull, NULL, 0, &parameter_desc);
```

```
the_alias = NULL;

    /* get the target application's creator */
err = FSpGetFInfo(document, &fndrInfo);
if (err != noErr) goto launch_the_document_termination;

    /* call the remap function if applicable */
app_creator_bytes = fndrInfo.fdCreator;
if (remap != NULL)
    remap(document, fndrInfo.fdType, &app_creator_bytes);

/* create an open documents Apple event */
err = AECreateDesc(typeApplSignature, (Ptr)target,
    sizeof(OSType), &target_desc);
if (err != noErr) goto launch_the_document_termination;
err = AECreateAppleEvent(kCoreEventClass, kAEOpenDocuments,
    &target_desc, kAutoGenerateReturnID, kAnyTransactionID,
        &the_apple_event);
if (err != noErr) goto launch_the_document_termination;

/* create a one element list of files to send in the event */
err = AECreateList(NULL, 0, false, &files_list);
if (err != noErr) goto launch_the_document_termination;
err = NewAlias(NULL, document, &the_alias);
if (err != noErr) goto launch_the_document_termination;
HLock((Handle) the_alias);
err = AECreateDesc(typeAlias, (Ptr) (*the_alias),
    GetHandleSize((Handle) the_alias), &file_desc);
HUnlock((Handle) the_alias);
if (err != noErr) goto launch_the_document_termination;
err = AEPutDesc(&files_list, 0, &file_desc);
if (err != noErr) goto launch_the_document_termination;

/* add the file list to the open documents event */
err = AEPutParamDesc(&the_apple_event, keyDirectObject,
    &files_list);
if (err != noErr) goto launch_the_document_termination;

/* send the Apple event */
err = AESend(&the_apple_event, &the_reply, kAENoReply,
        kAENormalPriority, kNoTimeOut, NULL, NULL);
```

```
/* if the target could not be found */
if (err == connectionInvalid) { /* no such app running? */
    HVolumeParam vol_pb;
    DTPBRec desktop_pb;
    FSSpec application;
    GetVolParmsInfoBuffer volinfo;
    HIOParam param_pb;
        /* search desktop files on local volumes first */
    param_pb.ioNamePtr = NULL;
    param_pb.ioBuffer = (Ptr) &volinfo;
    param_pb.ioReqCount = sizeof(volinfo);
    err = fnfErr; /* default return value */
    vol_pb.ioNamePtr = NULL;
    for (vol_pb.ioVolIndex = 1;
        PBHGetVInfoSync((HParmBlkPtr) &vol_pb) == noErr;
        vol_pb.ioVolIndex++) {
            param_pb.ioVRefNum = vol_pb.ioVRefNum;
            if (PBHGetVolParmsSync((HParmBlkPtr)&param_pb) == noErr
            && volinfo.vMServerAdr == 0) {
                desktop_pb.ioCompletion = NULL;
                desktop_pb.ioVRefNum = vol_pb.ioVRefNum;
                desktop_pb.ioNamePtr = NULL;
                desktop_pb.ioIndex = 0;
                if (PBDTGetPath(&desktop_pb) == noErr) {
                    desktop_pb.ioFileCreator = app_creator_bytes;
                    desktop_pb.ioNamePtr = application.name;
                    if (PBDTGetAPPLSync(&desktop_pb) == noErr) {
                        application.vRefNum = vol_pb.ioVRefNum;
                        application.parID = desktop_pb.ioAPPLParID;
                        err = noErr;
                        break;
                    }
                }
            }
        }
            /* if this fails, search remove volumes if allowed... */
        if (err != noErr && use_remote_apps)
            for (vol_pb.ioVolIndex = 1;
                PBHGetVInfoSync((HParmBlkPtr) &vol_pb) == noErr;
                vol_pb.ioVolIndex++) {
```

```
                    param_pb.ioVRefNum = vol_pb.ioVRefNum;
                    if (PBHGetVolParmsSync((HParmBlkPtr) &param_pb) ==
                                                                    noErr
                    && volinfo.vMServerAdr != 0) {
                        desktop_pb.ioCompletion = NULL;
                        desktop_pb.ioVRefNum = vol_pb.ioVRefNum;
                        desktop_pb.ioNamePtr = NULL;
                        desktop_pb.ioIndex = 0;
                        if (PBDTGetPath(&desktop_pb) == noErr) {
                            desktop_pb.ioFileCreator = app_creator_bytes;
                            desktop_pb.ioNamePtr = application.name;
                            if (PBDTGetAPPLSync(&desktop_pb) == noErr) {
                                application.vRefNum = vol_pb.ioVRefNum;
                                application.parID =
                                                desktop_pb.ioAPPLParID;
                                err = noErr;
                                break;
                            }
                        }
                    }
                }
                if (err != noErr) goto launch_the_document_termination;

            /* coerce the apple event to app parameters */
        err = AECoerceDesc(&the_apple_event, typeAppParameters,
                            &parameter_desc);
        if (err != noErr) goto launch_the_document_termination;
                /* launch the application */
            launch_pb.launchBlockID = extendedBlock;
            launch_pb.launchEPBLength = extendedBlockLen;
            launch_pb.launchFileFlags = 0;
            launch_pb.launchControlFlags =
                launchContinue + launchNoFileFlags;
            launch_pb.launchAppSpec = &application;
            HLock((Handle) parameter_desc.dataHandle);
            launch_pb.launchAppParameters =
                (AppParametersPtr) (*parameter_desc.dataHandle);
            err = LaunchApplication(&launch_pb);
            HUnlock((Handle) parameter_desc.dataHandle);
    }
    launch_the_document_termination:
```

```
            /* clean up, and go.. */
        if (the_alias != NULL) DisposeHandle((Handle) the_alias);
        AEDisposeDesc(&parameter_desc);
        AEDisposeDesc(&target_desc);
        AEDisposeDesc(&file_desc);
        AEDisposeDesc(&files_list);
        AEDisposeDesc(&the_apple_event);
        AEDisposeDesc(&the_reply);
        return err;
    }


/* sample creator mapping function. The function simply tells
    LaunchTheDocument that every document of type 'TEXT' should be
    opened using MPW Shell */
void MyMapCreator(FSSpec *document, OSType type, OSType *creator) {
    if (type == 'TEXT')
        *creator = 'BOBO'; /* open all text files with ClarisWorks,
                              creator = 'BOBO' */
}

    #ifdef powerc
    QDGlobals qd;
    #endif

void main(void) {
    SFTypeList typeList;
    StandardFileReply reply;

    InitGraf(&qd.thePort);
    InitFonts();
    InitWindows();
    TEInit();
    InitMenus();
    InitDialogs(0);
    InitCursor();

    StandardGetFile(NULL, -1, typeList, &reply);

    if (reply.sfGood)
```

```
        LaunchTheDocument(&reply.sfFile, false, MyMapCreator);

    }
```

File SendOpen.r containing Rez source for the `'SIZE'` resource. Because you're working with Apple events, you must include a SIZE resource with the `isHighLevelEventAware` bit set:

```
resource 'SIZE' (-1, purgeable) {
        reserved,
        acceptSuspendResumeEvents,
        reserved,
        canBackground,
        multiFinderAware,
        backgroundAndForeground,
        dontGetFrontClicks,
        ignoreChildDiedEvents,
        is32BitCompatible,
        isHighLevelEventAware,
        localAndRemoteHLEvents,
        isStationeryAware,
        dontUseTextEditServices,
        reserved,
        reserved,
        reserved,
        524288,
        524288
};
```

Summary

Launching an application with a document under System 7 requires the use of the AppleEvent Manager to either directly send an open document event to an existing application or indirectly send an open document event to an application as part of its launch sequence. This Technote describes how you can do this in your applications and provides a simple example of how you can use the technique.

Further References

- "Creating And Sending Apple Events" (Chapter 5), *Inside Macintosh: Interapplication Communication*
- "Handling the Required Apple Events" (4-11), *Inside Macintosh: Interapplication Communication*
- "Desktop Manager" (Chapter 9), *Inside Macintosh: More Macintosh Toolbox*
- "Obtaining Volume Information" (2-145), *Inside Macintosh: Files*
- "Alias Manager" (Chapter 4), *Inside Macintosh: Files*

Acknowledgements

Thanks to Eric Anderson, Nitin Ganatra, and Bob Wambaugh.