```
/*****************************************************************************

    FnAE.cp

*****************************************************************************/

/*
    You need to make minor modifications to the routines in this
    library to suit your needs.   Areas which you need to look at are
    noted with comments preceded with exclamation marks (!!).   When
    using AppleEvents in application, be sure to set program flag
    HighLevelEvent-Aware to TRUE so system actually sends AppleEvents
    to your application.

    Functions Include:

      FnAE_InitAE                 Installs AE procedures
      FnAE_DoHighLevelEvent     Processes AE
      FnAE_GotRequiredParams   Gets parameters from AE
      FnAE_OpenApp                Open application AE: core suite
      FnAE_OpenDoc                Open document AE: core suite
      FnAE_PrintDoc               Print document AE: core suite
      FnAE_Quit                   Quit application AE: core suite
      FnAE_SendOpenAE             Send Open AE to any file or application

      // Semi-private
      FnAE_FindProcess
*/

#include <AppleEvents.h>
#include <Processes.h>
#include <Aliases.h>

#define kFinderSig              'FNDR'
#define kAEFinderEvents       'FNDR'
#define kSystemType             'MACS'
#define kAEOpenSelection      'sope'
#define keySelection            'fsel'


// Prototypes

void    FnAE_InitAE( void );
void    FnAE_DoHighLevelEvent(   EventRecord        *theEvent );
OSErr   FnAE_GotRequiredParams( const AppleEvent *theEvent );
pascal OSErr FnAE_OpenApp(       AppleEvent          *theEvent,
                                 AppleEvent          *reply,
                                 long                refCon );
pascal OSErr FnAE_OpenDoc(       AppleEvent          *theEvent,
                                 AppleEvent          *reply,
                                 long                refCon );
pascal OSErr FnAE_PrintDoc(      AppleEvent          *theEvent,
                                 AppleEvent          *reply,
                                 long                refCon );
pascal OSErr FnAE_Quit(          AppleEvent          *theEvent,
                                 AppleEvent          *reply,
                                 long                refCon );
OSErr   FnAE_SendOpenAE(         FSSpec              *theDoc );

OSErr   FnAE_FindProcess( OSType                typeToFind,
                          OSType                creatorToFind,
                          ProcessSerialNumber* processSN );
```

```
extern void FnErr_DisplayStr( Str255 s1,
                              Str255 s2,
                              Str255 s3,
                              Str255 s4,
                              int    quitFlag );

/********** InitAE */

void FnAE_InitAE( void )

    OSErr    errCode;

    errCode = AEInstallEventHandler( kCoreEventClass,
                                     kAEOpenApplication,
                                     (AEEventHandlerProcPtr)FnAE_OpenApp,
                                     0,
                                     false);

    errCode = AEInstallEventHandler( kCoreEventClass,
                                     kAEOpenDocuments,
                                     (AEEventHandlerProcPtr)FnAE_OpenDoc,
                                     0,
                                     false);

    errCode = AEInstallEventHandler( kCoreEventClass,
                                     kAEPrintDocuments,
                                     (AEEventHandlerProcPtr)FnAE_PrintDoc,
                                     0,
                                     false);

    errCode = AEInstallEventHandler( kCoreEventClass,
                                     kAEQuitApplication,
                                     (AEEventHandlerProcPtr)FnAE_Quit,
                                     0,
                                     false);



/********** DoHighLevelEvent */

void FnAE_DoHighLevelEvent( EventRecord *theEvent )

    OSErr    errCode;
    /*
        Check to see if the Apple event is of a user-defined class;
        if so, handle it.   Otherwise, call AEProcessAppleEvent.
    */
    errCode = AEProcessAppleEvent( theEvent );
    if( errCode == errAEEventNotHandled )

        // !!??   Acknowledge( CantHandleAEVTID );
        // FnErr_DisplayStrID( 903, FALSE );



/********** GotRequiredParams */

OSErr FnAE_GotRequiredParams( const AppleEvent *theEvent )

    OSErr    errCode;
    Size     actualSize;
```

```
    DescType returnedType;

    errCode = AEGetAttributePtr( theEvent,
                                 keyMissedKeywordAttr,
                                 typeWildCard,
                                 &returnedType,
                                 NULL,
                                 0,
                                 &actualSize);
    if (errCode == errAEDescNotFound)

        // no parameters => no error
        errCode = noErr;

    else if (errCode == noErr)

        // got a parameter => it wasn't handled
        errCode = errAEEventNotHandled;

    return (errCode);



/********** OpenApp */

pascal OSErr FnAE_OpenApp( AppleEvent *theEvent,
                           AppleEvent *reply,
                           long        refCon )

    OSErr      errCode;

    errCode = FnAE_GotRequiredParams( theEvent );
    if (errCode == noErr)

        // !! call open application procedure

        // !! define prototype
        extern void NewWindow_( void );

        // !! now call procedure
        NewWindow_();

    return (errCode);



/********** OpenDoc */

pascal OSErr FnAE_OpenDoc( AppleEvent *theEvent,
                           AppleEvent *reply,
                           long        refCon )

    OSErr          errCode;
    OSErr          ignoreErr;
    AEDescList     docList;
    long           itemsInList;
    long           index;
    AEKeyword      keyword;
    DescType       returnedType;
    Size           actualSize;
    FSSpec         myFSS;
    short          wdRefNum;
```

```
    errCode = AEGetParamDesc( theEvent,
                              keyDirectObject,
                              typeAEList,
                              &docList );
    if( errCode == noErr )

        errCode = FnAE_GotRequiredParams( theEvent );
        if( errCode == noErr )

            errCode = AECountItems( &docList, &itemsInList );
            if( errCode == noErr )

                for( index = 1; index <= itemsInList; index++ )

                    errCode = AEGetNthPtr( &docList,
                                           index,
                                           typeFSS,
                                           &keyword,
                                           &returnedType,
                                           (Ptr)&myFSS,
                                           sizeof(myFSS),
                                           &actualSize );
                    if( errCode == noErr )

                        // !! open window and file
                        /***
                        errCode = OpenWD( myFSS.vRefNum,
                                          myFSS.parID,
                                          0,
                                          &wdRefNum );
                        if( errCode == noErr )

                            OpenDoc( myFSS.name, wdRefNum );

                        ***/



        ignoreErr = AEDisposeDesc( &docList );

    return (errCode);


/********** PrintDoc */

pascal OSErr FnAE_PrintDoc( AppleEvent *theEvent,
                            AppleEvent *reply,
                            long          refCon )

    OSErr      errCode;

    errCode = FnAE_GotRequiredParams( theEvent );
    if( errCode == noErr )

        // !!??   Acknowledge( CantHandleAEVTID );

    return( errCode );
```

```
/********** Quit */

pascal OSErr FnAE_Quit( AppleEvent *theEvent,
                        AppleEvent *reply,
                        long          refCon )

    OSErr      errCode;

    errCode = FnAE_GotRequiredParams( theEvent );
    if( errCode == noErr )

        // !! call application quit procedure

        // !! define prototype
        extern void Quit_( void );

        // !! now call procedure
        Quit_();

    return( errCode );



/********** SendOpenAE */

OSErr FnAE_SendOpenAE( FSSpec *theDoc )

    AppleEvent              aeEvent;            // the event to create
    AEDesc                  myAddressDesc;   // descriptors for the AE
    AEDesc                  aeDirDesc;
    AEDesc                  listElem;
    AEDesc                  fileList;          // our list
    FSSpec                  dirSpec;
    AliasHandle             dirAlias;          // alias to directory
    AliasHandle             fileAlias;         // alias of the file itself
    ProcessSerialNumber process;            // the finder's psn
    OSErr                   myErr;             // duh

    // Get the psn of the Finder and create the target address for AE
    if(FnAE_FindProcess(kFinderSig,kSystemType,&process))

        FnErr_DisplayStr(
            "\pThe Finder must be running in order ",
            "\pto be able to send Apple Event.",
            "\p",
            "\p",
            FALSE );
        return procNotFound;

    myErr = AECreateDesc( typeProcessSerialNumber,
                          (Ptr)&process,
                          sizeof(process),
                          &myAddressDesc );
    if(myErr)
        return myErr;

    // Create an empty AppleEvent
    myErr = AECreateAppleEvent( kAEFinderEvents,
                                kAEOpenSelection,
                                &myAddressDesc,
                                kAutoGenerateReturnID,
```

```
                                    kAnyTransactionID,
                                    &aeEvent );
    if(myErr)
        return myErr;

    // Make an FSSpec and alias for the parent folder, and for the file
    FSMakeFSSpec(theDoc->vRefNum,theDoc->parID,nil,&dirSpec);
    NewAlias(nil,&dirSpec,&dirAlias);
    NewAlias(nil,theDoc,&fileAlias);

    // Create the file list.
    myErr=AECreateList(nil,0,false,&fileList);
    if(myErr)
        return myErr;

    // Create the folder descriptor
    HLock((Handle)dirAlias);
    AECreateDesc(typeAlias, (Ptr) *dirAlias, GetHandleSize
                ((Handle) dirAlias), &aeDirDesc);
    HUnlock((Handle)dirAlias);
    DisposHandle((Handle)dirAlias);

    if((myErr = AEPutParamDesc(&aeEvent,keyDirectObject,&aeDirDesc)) ==
        noErr)

        AEDisposeDesc(&aeDirDesc);
        HLock((Handle)fileAlias);

        AECreateDesc(typeAlias, (Ptr)*fileAlias,
            GetHandleSize((Handle)fileAlias), &listElem);
        HUnlock((Handle)fileAlias);
        DisposHandle((Handle)fileAlias);
        myErr = AEPutDesc(&fileList,0,&listElem);

    if(myErr)
        return myErr;
    AEDisposeDesc(&listElem);

    myErr = AEPutParamDesc(&aeEvent,keySelection,&fileList);
    if(myErr)
        return myErr;

    myErr = AEDisposeDesc(&fileList);

    myErr = AESend(&aeEvent, nil,
            kAENoReply+kAEAlwaysInteract+kAECanSwitchLayer,
            kAENormalPriority, kAEDefaultTimeout, nil, nil);
    AEDisposeDesc(&aeEvent);

    return noErr;



/********** FindProcess */

OSErr FnAE_FindProcess( OSType                  typeToFind,
                        OSType                  creatorToFind,
                        ProcessSerialNumberPtr  processSN )

    ProcessInfoRec  tempInfo;
    FSSpec          procSpec;
    Str31           processName;
```

```
    OSErr           myErr = noErr;
    Boolean         done;

    // start at the beginning of the process list
    processSN->lowLongOfPSN = kNoProcess;
    processSN->highLongOfPSN = kNoProcess;

    // initialize the process information record
    tempInfo.processInfoLength = sizeof(ProcessInfoRec);
    tempInfo.processName = (StringPtr)&processName;
    tempInfo.processAppSpec = &procSpec;

    done = FALSE;
    while( done == FALSE )

        myErr = GetNextProcess( processSN );
        if( myErr == noErr )
            GetProcessInformation( processSN, &tempInfo );
        else
            done = TRUE;

        if( tempInfo.processSignature == creatorToFind ||
            tempInfo.processType == typeToFind )
            done = TRUE;


    return(myErr);


// End of File
```