Writing Apple Event Handlers

For each Apple event your application supports, you must provide a function called an Apple event handler. The **AEProcessAppleEvent** function calls one of your Apple event handlers when it processes an Apple event. Your Apple event handlers should perform any action requested by the Apple event, add parameters to the reply Apple event if appropriate, and return a result code.

The **Apple Event Manager** uses dispatch tables to route Apple events to the appropriate Apple event handler. You must supply an Apple event handler for each entry in your application's Apple event dispatch table. Each handler must be a function that uses this syntax:

```
OSErr MyEventHandler(AppleEvent *theAppleEvent, AppleEvent *reply,
                     long handlerRefcon);
```

The parameter *theAppleEvent* is the Apple event to handle. Your handler uses **Apple Event Manager** functions to extract any parameters and attributes from the Apple event and then performs the necessary processing. The *reply* parameter is the default reply provided by the**Apple Event Manager**. The *handlerRefcon* parameter is the reference constant stored in the Apple event dispatch table entry for the Apple event. Your handler can ignore this parameter if your application does not use the reference constant.

After extracting all known parameters from the Apple event, every handler should determine whether the Apple event contains any further required parameters. Your handler can check that it retrieved all the required parameters by checking to see if the keyMissedKeywordAttr attribute exists. If the attribute exists, then your handler has not retrieved all the required parameters. If additional required parameters exist, then your handler should immediately return an error. If the attribute does not exist, then the Apple event does not contain any more required parameters.

The following program shows a function that checks for a keyMissedKeywordAttr attribute. A handler calls this function after getting all the parameters it knows about from an Apple event.

```
// Assuming inclusion of <MacHeaders>

#include <AppleEvents.h>
OSErr  MyGotRequiredParams (AppleEvent *theAppleEvent);

OSErr  MyGotRequiredParams (AppleEvent *theAppleEvent)
{
    DescType   returnedType;
    Size       actualSize;
    OSErr      myErr;

    myErr = AEGetAttributePtr(theAppleEvent, keyMissedKeywordAttr,
                typeWildCard, &returnedType,
                nil, 0, &actualSize);

    if (myErr == errAEDescNotFound)   // you got all the required
                                      //parameters
        return noErr;
```

```
    else
        if (myErr == noErr)  // you missed a required parameter
            return errAEParamMissed;
        else                 // the call to AEGetAttributePtr failed
            return myErr;
}
```

This code in uses the **AEGetAttributePtr** function to get the keyMissedKeywordAttr attribute. This attribute contains the first required parameter, if any, that your handler didn't retrieve. If **AEGetAttributePtr** returns the errAEDescNotFound result code, the Apple event doesn't contain a keyMissedKeywordAttr attribute. If the Apple event doesn't contain this attribute, then your handler has extracted all of the required parameters.

If the **AEGetAttributePtr** function returns noErr as the result code, then the attribute does exist, meaning that your handler has not extracted all of the required parameters. In this case, your handler should return an error and not process the Apple event.

The first remaining required parameter is specified by the data of the keyMissedKeywordAttr attribute. If you want this data returned, specify a buffer to hold the data. Otherwise, specify NIL as the buffer and 0 as the size of the buffer. If you specify a buffer to hold the data, you can check the value of the *actualSize* parameter to see if the data is larger than the buffer you allocated.