Installing Apple Event Handlers

When your application receives an Apple event, use the **AEProcessAppleEvent** function to retrieve the data buffer of the event and to route the Apple event to the appropriate Apple event handler in your application. Your application supplies an Apple event dispatch table to provide a mapping between the Apple events your application supports and the Apple event handlers provided by your application.

To install entries into your application's Apple event dispatch table, use the **AEInstallEventHandler** function. You usually install entries for all of the Apple events that your application accepts into your application's Apple event dispatch table.

For each Apple event your application supports, you should install entries in your Apple event dispatch table that specify

- the event class of the Apple event
- the event ID of the Apple event
- the address of the Apple event handler for the Apple event
- a reference constant

You provide this information to the **AEInstallEventHandler** function. In addition, you indicate to the **AEInstallEventHandler** function whether the entry should be added to your application's Apple event dispatch table or the system Apple event dispatch table.

The **system Apple event dispatch table** is a table in the system heap that contains handlers that are available to all applications and processes running on the same computer. The handlers in your application's Apple event dispatch table are available only to your application. If **AEProcessAppleEvent** cannot find a handler for the Apple event in your application's Apple event dispatch table, it looks in the system Apple event dispatch table for a handler. If it doesn't find a handler there either, it returns the errAEEventNotHandled result code.

The following program illustrates how to add entries for the required Apple Events to your application's Apple event dispatch table. See Open Application Event, Open Documents Event, Print Documents Event and Quit Application Event for code examples of event handlers for each type of event.

```
// Adding entries for the required Apple Events to an
// application's Apple event dispatch table

// Assuming inclusion of <MacHeaders>

#include <AppleEvents.h>

pascal  OSErr  MyHandleOAPP(AppleEvent *theAppleEvent, AppleEvent *reply,
                                        long handlerRefcon);
pascal  OSErr  MyHandleODOC(AppleEvent *theAppleEvent, AppleEvent *reply,
                                        long handlerRefcon);
```

```
pascal  OSErr  MyHandlePDOC(AppleEvent *theAppleEvent, AppleEvent *reply,
                                              long handlerRefcon);
pascal  OSErr  MyHandleQUIT(AppleEvent *theAppleEvent, AppleEvent *reply,
                                              long handlerRefcon);

void DoError (OSErr myErr);

OSErr myErr;

myErr = AEInstallEventHandler(kCoreEventClass,
                                              kAEOpenApplication,
                                              &MyHandleOAPP, 0, FALSE);
if (myErr)
    DoError(myErr);

myErr = AEInstallEventHandler(kCoreEventClass,
                                              kAEOpenDocuments,
                                              &MyHandleODOC, 0, FALSE);
if (myErr)
    DoError(myErr);

myErr = AEInstallEventHandler(kCoreEventClass,
                                              kAEPrintDocuments,
                                              &MyHandlePDOC, 0, FALSE);
if (myErr)
    DoError(myErr);

myErr = AEInstallEventHandler(kCoreEventClass,
                                              kAEQuitApplication,
                                              &MyHandleQUIT, 0, FALSE);
if (myErr)
    DoError(myErr);
```

The code above creates an entry for all required Apple Events in the Apple event dispatch table. The first entry creates an entry for the Open Application event. The entry indicates the event class and event ID of the Open Application event and the address of the handler for that event and specifies 0 as the reference constant. This entry is installed into the application's Apple event dispatch table.

The reference constant is passed to your handler by the **Apple Event Manager** each time your handler is called. Your application can use this reference constant for any purpose. If your application doesn't use the reference constant, use 0 as the value.

The last parameter to the **AEInstallEventHandler** function is a Boolean value that determines whether the entry is added to the system Apple event dispatch table or to your application's Apple event dispatch table. To add the entry to your application's dispatch table, use FALSE as the value of this parameter. If you specify TRUE, the entry is added to the system Apple event dispatch table.

If you add a handler to the system Apple event dispatch table, the handler that you specify must reside in the system heap. If there was already an entry in the system Apple event dispatch table for the same event class and event ID, it is replaced. Therefore, if there is an entry in the system Apple event dispatch table for the same event class and event ID, you should chain it to your system handler.

Note: When an application calls a system Apple event handler, the A5 register is set up for the calling application. For this reason, if you provide a system Apple event handler, it should never use A5 global variables or anything that depends on a particular context; otherwise, the application that calls the system handler may crash.

For any entry in your Apple event dispatch table, you can specify a wildcard value for the event class, event ID, or both. You specify a wildcard by supplying the typeWildCard constant when installing an entry into the Apple event dispatch table. A wildcard value matches all possible values.

For example, if you specify an entry with the typeWildCard event class and the kAEOpenDocuments event ID, the **Apple Event Manager** dispatches Apple events of any event class and an event ID of kAEOpenDocuments to the handler for that entry.

If you specify an entry with the kCoreEventClass event class and the typeWildCard event ID, the **Apple Event Manager** dispatches Apple events of the kCoreEventClass event class and any event ID to the handler for that entry.

If you specify an entry with the typeWildCard event class and the typeWildCard event ID, the **Apple Event Manager** dispatches all Apple events of any event class and any event ID to the handler for that entry.

If the **AEProcessAppleEvent** function cannot find a handler for an Apple event in either the application's Apple event dispatch table or the system Apple event dispatch table, it returns the result code errAEEventNotHandled to the Apple event server. If the client is waiting for a reply, **AESend** also returns this result code as its function result.

If your application supports the **Edition Manager**, you should also add entries to your application's Apple event dispatch table for the Apple events that your application receives from the **Edition Manager**. The following program shows how to add entries for these Apple events to your application's Apple event dispatch table.

```
// Adding entries for Apple events sent by the Edition
// Manager to an application's Apple event dispatch table

// Assuming inclusion of <MacHeaders>

#include <AppleEvents.h>
#include <Editions.h>

pascal  OSErr  MyHandleSectionReadEvent(AppleEvent *theAppleEvent,
               AppleEvent *reply, long handlerRefcon);
pascal  OSErr  MyHandleSectionWriteEvent(AppleEvent *theAppleEvent,
               AppleEvent *reply, long handlerRefcon);
pascal  OSErr  MyHandlePDOC(AppleEvent *theAppleEvent, AppleEvent *reply,
                    long handlerRefcon);
pascal  OSErr  MyHandleQUIT(AppleEvent *theAppleEvent, AppleEvent *reply,
                    long handlerRefcon);
void DoError (OSErr myErr);

OSErr  myErr;

myErr = AEInstallEventHandler(sectionEventMsgClass,
                                        sectionReadMsgID,
                                        &MyHandleSectionReadEvent,
                                        0, FALSE);
if (myErr)
    DoError(myErr);

myErr = AEInstallEventHandler(sectionEventMsgClass,
                                        sectionWriteMsgID,
                                        &MyHandleSectionWriteEvent,
                                        0, FALSE);
if (myErr)
    DoError(myErr);

myErr = AEInstallEventHandler(sectionEventMsgClass,
                                        sectionScrollMsgID,
                                        &MyHandleSectionScrollEvent,
                                        0, FALSE);
if (myErr)
    DoError(myErr);
```