Responding to Events

You can identify high-level events by the value in the *what* field of the EventRecord. The *message* and where fields further classify the type of high-level event. Your application can choose to recognize as many events as are appropriate. Some high-level events may be fully specified by their EventRecord only, while others may include additional information in an optional buffer. To get that additional information or to find the sender of the event, use the **AcceptHighLevelEvent** function.

Note: To respond to an Apple event, use the **Apple Event Manager**, as described in the **Apple Event Manager** description.

The following program illustrates how to call **AcceptHighLevelEvent**. In general, you cannot know in advance how big the optional data buffer is, so you can allocate a zero-length buffer and then resize it if the call to **AcceptHighLevelEvent** returns the error bufferIsSmall.

Accepting a high-level event

```
// Assumes inclusion of <MacHeaders>

#include <EPPC.h>

void DoError            (OSErr myErr);

TargetID                myTarg;
unsigned long           myRefCon;
Ptr myBuff;
unsigned long           myLen;
OSErr                   myErr;

myLen = 0;    // Start with a 0-byte buffer
myBuff     = nil;
myErr = AcceptHighLevelEvent(&myTarg, &myRefCon, myBuff,
            &myLen);

if (myErr == bufferIsSmall)
    {
    myBuff= NewPtr(myLen);  // Get new pointer
    myErr = AcceptHighLevelEvent(&myTarg, &myRefCon, myBuff,
            &myLen);
    }

if (myErr)
    DoError(myErr);
```

The ID of the sender of the event is returned in the first parameter, which is a targetID record. You can inspect the fields of that record to determine which application sent the event. That record also contains the session reference number that identifies this communication as well as the port name and port location of the sender. If the high-level event requires that you return information, you can use the value returned in the *sender* parameter to send an event back to the requesting application.

The *buffer* parameter points to any additional data associated with the event. Any data in the additional buffer is defined by the particular high-level event. On input, the *length* parameter contains the size of the buffer. If no error occurs, on output the *length* parameter contains the size of the *message* accepted. If the error bufferIsSmall occurs, the *length* parameter contains the size of the *message* yet to be received. The reference constant parameter is a unique number your application can use to identify communication associated with this event.