Replying to an Apple Event

Your handler routine for a particular Apple event is responsible for performing the action requested by the Apple event, and can optionally return data in a reply Apple event. The **Apple Event Manager** passes a default reply Apple event to your handler. The default reply Apple event has no parameters when it is passed to your handler. Your handler can add parameters to the reply Apple event. If the client application requested a reply, the **Apple Event Manager** returns the reply Apple event to the client.

The reply Apple event is identified by the kCoreEventClass event class and by the kAEAnswer event ID.

When your handler finishes processing an Apple event, it returns a result code to **AEProcessAppleEvent**. The **AEProcessAppleEvent** function returns this result code as its function result. If your handler returns a nonzero result code, the **Apple Event Manager** also returns this result code to the client application by putting the result code into a keyErrorNumber parameter for the reply Apple event. The client can check for the existence of this parameter to determine whether the handler performed the requested action.

The client application specifies whether it wants a reply Apple event or not by specifying flags (represented by constants) in the *sendMode* parameter of the **AESend** function.

If the client specifies the kAEWaitReply flag in the *sendMode* parameter, the **AESend** function does not return until the timeout expires or the server returns a reply. When the server returns a reply, the *reply* parameter to **AESend** contains the reply Apple event that your handler returned to the **AEProcessAppleEvent** function.

If the client specified the kAEQueueReply flag, the client receives the reply event in its normal processing of other events.

If the client specified the kAENoReply flag, your handler may return a reply Apple event to **AEProcessAppleEvent**, but this reply is not returned to the client.

Your handler routine should always set its function result to noErr if it successfully handles the Apple event or to a nonzero result code if an error occurs. The Apple Event Manager automatically adds any nonzero result code that your handler returns to a keyErrorNumber parameter in the reply Apple event. In addition to returning a result code, your handler can also return an error string in the keyErrorString parameter of the reply Apple event. Your handler should provide meaningful text in the keyErrorString parameter, so that the client can display this string to the user if desired.

The following program shows how to add the keyErrorString parameter to the reply Apple event.

```
// Adding the keyErrorString parameter to the reply Apple event

// Assuming inclusion of <MacHeaders>
```

```
#include <AppleEvents.h>

pascal OSErr MyHandler (AppleEvent *theAppleEvent, AppleEvent *reply,
                        long handlerRefcon);

pascal OSErr MyHandler (AppleEvent *theAppleEvent, AppleEvent *reply,
                        long handlerRefcon)
{
    OSErr  myErr;
    unsigned char *errStr;

    // if an error occurs when handling an Apple event,
    // set the function result and error string accordingly
    if (myErr) {  // myErr is the result code to be returned--

    // the Apple Event Manager adds this result
    // code to the reply Apple event as the
    //  keyErrorNumber parameter

        if (reply->dataHandle != nil) {
            // add error string parameter to the default reply
            errStr = "\pWhy error occurred";   // strings should
                                // normally  be stored in resources
            myErr = AEPutParamPtr(reply, keyErrorString, typeChar,
                                (Ptr)&errStr[1], errStr[0]);
        }
    }
    return myErr;
}
```

If your handler needs to return data to the client, it can add parameters to the reply Apple event. For example, the next program shows how a handler for the Multiply event (an imaginary Apple event that asks the server to multiply two numbers) might return the results of the multiplication to the client.

```
// Adding parameters to the reply Apple event

// Assuming inclusion of <MacHeaders>
#include <AppleEvents.h>

pascal OSErr MyMultHandler (AppleEvent *theAppleEvent,
                    AppleEvent *reply, long handlerRefcon);
OSErr MyDoMultiply(AppleEvent *theAppleEvent,long number1,
                    long number2, long *replyResult);

pascal OSErr MyMultHandler (AppleEvent *theAppleEvent,
                    AppleEvent *reply, long handlerRefcon)
{

    OSErr  myErr, returnErr;
    long    number1,
            number2;
    long    replyResult;
```

```
    Size     actualSize;
    DescType   returnedType;
    unsigned char *errStr;

    // Get the numbers to multiply from the  parameters  of  the
    // Apple event; put the numbers in  the  number1  and  number2
    // variables and then  perform  the  requested  multiplication


    myErr = MyDoMultiply(theAppleEvent, number1, number2,
                    &replyResult);

    // Return result of the multiplication in the reply Apple event

    if ((reply->dataHandle) && (!myErr)) {

        myErr = AEPutParamPtr(reply, keyDirectObject,
                        typeLongInteger, (Ptr) &replyResult,
                        sizeof(replyResult));
    }
    // If an error occurs, set the function result and error string
    // accordingly
    if (myErr) {
        errStr = "\pReason why error occurred";
        myErr = AEPutParamPtr (reply, keyErrorString, typeChar,
                        (Ptr) &errStr[1],
        errStr[0]);
    }
    return myErr;
}
```