Dealing With Timeouts

When your application calls **AESend** and chooses to wait for the server to handle the Apple event, it can also specify the maximum amount of time that it is willing to wait for a response. You can specify a timeout value in the *timeOutInTicks* parameter to **AESend**. You can either specify a particular length of time, in ticks, that your application is willing to wait, or you can specify the kNoTimeOut constant or the kAEDefaultTimeout constant.

Use the kNoTimeOut constant to indicate that your application is willing to wait forever for a response from the server. You should use this value only if your application is guaranteed that the server will respond in a reasonable amount of time. You should also implement a method of checking if the user wants to cancel. The idle function that you specify as a parameter to **AESend** should check the event queue for any instances of Command-period and immediately return TRUE as its function result if it finds a request to cancel in the event queue.

Use the kAEDefaultTimeout constant if you want the **Apple Event Manager** to use a default value for the timeout value. The **Apple Event Manager** uses a timeout value of about one minute if you specify this constant.

Note that if you set the kAEWaitReply flag and the server doesn't have a handler for the Apple event, **AESend** returns immediately with the errAEEventNotHandled result code.

If the server doesn't respond within the length of time specified by the timeout value, **AESend** returns the errAETimeout result code. This result code does not necessarily mean that the server failed to perform the requested action; it only means that the server did not complete processing within the specified time. The server might still be processing the Apple event, and it might still send a reply.

If the server finishes processing the Apple event sometime after the time specified in the timeout parameter has expired, it returns a reply Apple event to **AEProcessAppleEvent**. The **Apple Event Manager** then returns the reply to the client in the *reply* parameter that the client originally passed to the **AESend** function.

This means your application can continue to check the reply Apple event to see if the server has responded, even after the time expires. If the server has not yet sent the reply when the client attempts to extract data from the reply Apple event, the **Apple Event Manager** functions return the errAEReplyNotArrived result code. Once the reply Apple event returns from the server, the client can extract the data in the reply.

Additionally, the server can determine the timeout value specified by the client by examining the keyTimeoutAttr attribute in the Apple event. You can use the value of this attribute as a rough estimate of how much time your handler has to respond. You can assume that your handler has less time to respond than the timeout value, because transmitting the Apple event uses some of the available time, as does transmitting the reply Apple event back to the client.

If your handler needs more time than is specified in the keyTimeoutAttr

attribute, you can reset the timer by using the **AEResetTimer** function. This function resets the timeout value of an Apple event to its starting value.