Interacting With the User

When your application receives an Apple event, it may need to interact with the user. For example, your application may need to display a dialog box asking for additional input or confirmation from the user. You must make sure that your application is in the foreground before interacting with the user. To do this, use the **AEInteractWithUser** function before actually interacting with the user. The **AEInteractWithUser** function checks the user interaction preferences set by the client application and the server application and, if user interaction is allowed, brings your application to the front (either directly or by posting a notification request) if it is not already in the front.

If both the client and server applications allow user interaction, **AEInteractWithUser** usually posts a notification request; **AEInteractWithUser** brings the server to the front after the user responds to the notification request. The **AEInteractWithUser** function can also bring the server application directly to the front-but only when doing so is in accordance with the principle of user control and if the client allows it.

Both the client and server specify their preferences for user interaction: the client specifies whether the server should be allowed to interact with the user, and the server specifies when it will allow user interaction while processing an Apple event.

An application that sends an Apple event indicates its preferences for how the server application should interact with the user by setting various flags in the *sendMode* parameter to **AESend**. The **Apple Event Manager** translates these flags into the corresponding flags in the keyInteractLevelAttr attribute of the Apple event, and sets them.

The server application sets its preferences by using the **AESetInteractionAllowed** function. This function lets your application specify whether it allows interaction with the user as a result of receiving an Apple event from itself; from itself and other processes on the local machine; or from itself, local processes, and processes from another computer on the network.

Your application calls the **AEInteractWithUser** function before interacting with the user. If **AEInteractWithUser** returns the noErr result code, then your application is currently in the front and your application is free to interact with the user. If **AEInteractWithUser** returns the errAENoUserInteraction result code, then the conditions didn't allow user interaction and your application should not interact with the user.

The client application sets user interaction preferences by setting flags in the *sendMode* parameter to the **AESend** function. The **Apple Event Manager** automatically adds the specified flags to the keyInteractLevelAttr attribute of the Apple event. These flags are represented by constants and are described here.

Flag	Description
kAENeverInteract	The server application should never interact with the user in response to this Apple event. If this flag is set, **AEInteractWithUser** does not bring the

server application to the foreground (this is the default when an Apple event is sent to a remote application).

kAECanInteract

The server application can interact with the user in response to this Apple event-by convention, if the user needs to supply information to the server. If this flag is set and the server allows interaction, **AEInteractWithUser** brings the server application to the foreground (this is the default when an Apple event is sent to a local application).

kAEAlwaysInteract

The server application can interact with the user in response to this Apple event-by convention, even if no information is needed from the user. If this flag is set and the server allows interaction, **AEInteractWithUser** brings the server application to the foreground. The **Apple Event Manager** does not distinguish between this flag and the kAECanInteract flag-distinguishing between them is the responsibility of the server application.

If the client application doesn't specify any of the user interface flags, the **Apple Event Manager** sets either the kAENeverInteract or the kAECanInteract flag in the keyInteractLevelAttr attribute of the Apple event, depending on the location of the server application. If the server application is on a remote computer, the **Apple Event Manager** sets the kAENeverInteract flag as the default. If the server application is on the local computer, the **Apple Event Manager** sets the kAECanInteract flag as the default.

In addition, the client application can set another flag in the *sendMode* parameter to **AESend** to request that the **Apple Event Manager** immediately bring the server application to the front (instead of posting a notification request**)–if** user interaction is allowed and if the user interface guidelines permit.

kAECanSwitchLayer

If both the client and server allow interaction and this flag is set, **AEInteractWithUser** brings the server directly to the foreground if adherence to the principle of user control allows. If the action would be contrary to this principle, **AEInteractWithUser** uses the **Notification Manager** to request that the user bring the server application to the foreground. If

both the client and server allow interaction and this flag is not set, **AEInteractWithUser** always uses the **Notification Manager** to request that the user bring the server application to the foreground.

When a server application calls **AEInteractWithUser**, the function first checks to see if the kAENeverInteract flag in the keyInteractLevelAttr attribute of the Apple event is set. (The **Apple Event Manager** sets this attribute according to the flags specified in the *sendMode* parameter of **AESend**.) If the kAENeverInteract flag is set, **AEInteractWithUser** immediately returns the errAENoUserInteraction result code. If the client specified kAECanInteract or kAEAlwaysInteract, **AEInteractWithUser** checks the server's preferences for user interaction.

The server sets its user interaction preferences by using the **AESetInteractionAllowed** function. You use this function to tell the **Apple Event Manager** the processes for which your application is willing to interact with the user.

myErr = **AESetInteractionAllowed(***level***);**

The *level* parameter is of type AEInteractAllowed.

You can specify one of these values for the interaction level.

Flag	Description
kAEInteractWithSelf	User interaction with your server application in response to an Apple event may be allowed only when the client application is your own application-that is, only when your application is sending the Apple event to itself.
kAEInteractWithLocal	User interaction with your server application in response to an Apple event may be allowed only if the client application is on the same computer as your application; this is the default if the **AESetInteractionAllowed** function is not used.
kAEInteractWithAll	User interaction with your server application in response to an Apple event may be allowed for any client application on any computer.

If the server application does not set the user interaction level, **AEInteractWithUser** uses kAEInteractWithLocal as the value.

If the application sent itself an Apple event (that is, the application is both the client and the server), **AEInteractWithUser** always allows user interaction. If the client application is a process on the local machine, and the server set the interaction level to the kAEInteractWithLocal or kAEInteractWithAll flag, then **AEInteractWithUser** allows user interaction. If the client is a process on a remote computer on the network, **AEInteractWithUser** allows user interaction only if the server specified

the kAEInteractWithAll flag for the interaction level. In all other cases, **AEInteractWithUser** does not allow user interaction.

When **AEInteractWithUser** allows user interaction (based on the client's and server's preferences), **AEInteractWithUser** brings the server application to the front-either directly or after the user responds to a notification request-and then returns a noErr result code.

If **AEInteractWithUser** cannot bring the server application to the front within the specified timeout value, **AEInteractWithUser** returns the errAETimeout result code.

Your application may want to provide the user with a method of setting the interaction level. For example, some users may not want to be interrupted while background processing of an Apple event occurs, or they may not want to respond to dialog boxes when your application is handling Apple events sent from another computer.

The following program illustrates the use of the **AEInteractWithUser** function. You call this function before your application displays a dialog box or otherwise interacts with the user when processing an Apple event. You specify a timeout value, a pointer to a **Notification Manager** record, and the address of an idle function as parameters to **AEInteractWithUser**.

```
// Using the AEInteractWithUser function

// Assuming inclusion of <MacHeaders>

#include <AppleEvents.h>

OSErr  InteractWithUser (void);
void        DoError (OSErr myErr);
void        DisplayMyDialogBox (void);
pascal  Boolean MyIdleFunction (EventRecord *theEvent, long *sleepTime,
                                                RgnHandle *mouseRgn);

OSErr InteractWithUser ()
{
    OSErr myErr;

    myErr = AEInteractWithUser (kAEDefaultTimeout, nil,
                                                &MyIdleFunction);

    if (!myErr)

        // The attempt to interact failed; do any error handling
        DoError(myErr);
    else
        // Interact with the user by displaying a dialog box
        // or by interacting in any other way that is necessary

        DisplayMyDialogBox ();
    return myErr;
}
```

You can set a timeout value, in ticks, in the first parameter to **AEInteractWithUser**. 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. You can also specify the kNoTimeOut constant if your application is willing to wait an indefinite amount of time for a response from the user. Usually you should provide a timeout value, so that your application can complete processing of the Apple event in a reasonable amount of time.

You can provide a pointer to a **Notification Manager** record in the second parameter, or you can specify NIL to use the default record provided by **AEInteractWithUser**. The **AEInteractWithUser** function only uses a **Notification Manager** record when user interaction is allowed and the kAECanSwitchLayer flag in the keyInteractLevelAttr attribute is not set.

The last parameter to **AEInteractWithUser** specifies an idle function provided by your application. Your idle function should handle any update events, null events, operating-system events, or activate events while your application is waiting to be brought to the front.