Apple Events

Apple events provide your application with a standard mechanism for communicating with other applications. You can use Apple events and the **Apple Event Manager** to

- respond to the required Apple Events (Open Application, Open Documents, Print Documents, and Quit Application) that are sent by the Finder
- respond to the Apple events sent by the **Edition Manager** and allow users to share data among documents created by multiple applications
- provide services to other applications. See **Responding to Apple Events**
- request services from other applications. See **Requesting Services Through Apple Events**

By supporting the required Apple Events, your application can take advantage of the more reliable launch and termination mechanisms built into system 7.0. You can also take advantage of the services provided by the **Edition Manager** by responding to the Apple events sent by the **Edition Manager**. These and additional core Apple events can be used by nearly all applications to communicate with system software or with other applications.

You can also support functional-area Apple events related to your application in order to provide services to other applications or to request services from other applications. Finally, if your application defines Apple events for all the actions that a user can perform, you can record user actions by generating the corresponding Apple event for each action, saving a copy of the Apple event, and then sending the Apple event to your own application for handling. Apple events that are recorded in this way can later be played back to automate tasks previously performed by the user.

To support Apple events in your application, you must

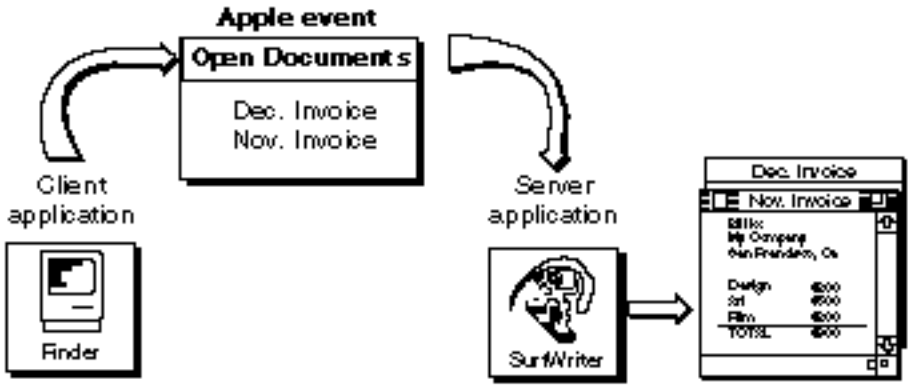
- decide which Apple events (in addition to the required ones) to support. See **Apple Event Types**.
- set bits in the 'SIZE' resource to indicate that your application supports high-level events. See **Accepting an Apple Event**.
- create an Apple event dispatch table. See **Writing Apple Event Handlers** and **Installing Apple Event Handlers**.
- include code to handle high-level events in your main event loop. See **Accepting an Apple Event**.
- handle the Apple events your application receives and wishes to support. See **Responding to Apple Events**.
- create the Apple events you wish your application to generate. See **Requesting Services Through Apple Events**.

Introduction to Apple Events

Applications typically use Apple events to request services from and provide services to other applications. Thus, the Open Documents event, sent by the Finder, requests that your application open specified documents. When your application supports this Apple event, it should respond by opening those documents in the manner that your application normally opens documents.

A transaction involving Apple events is initiated by a "client application", which sends an Apple event to request a service (for example, printing a list of files, spell-checking a list of words, or performing a numerical calculation). The application providing the service is called a "server application". These applications can reside on the same local computer or on remote computers connected to a network.

The following figure shows a common Apple event, the Open Documents event. You see that the Finder application is the client; it requests that the SurfWriter application open the documents named Dec. Invoice and Nov. Invoice. The SurfWriter application responds to the Finder's request by opening windows containing the specified documents.



An Open Documents event

The Finder is also the source application of the Open Documents event. A source application is one that sends an Apple event to another application or to itself. In the figure above, the SurfWriter application is the target application of the event. The target application is the one addressed to receive the Apple event. The terms *client application* and *source application* are not always synonymous, nor are the terms *server application* and *target application*. Typically, an Apple event client sends an Apple event requesting a service from an Apple event server; in this case, the server is the target application of the Apple event. The Apple event server may send back a different Apple event as a response-in which case, the client becomes the target of the responding Apple event.