

## t Access to Events <sup>\*158\*</sup>

This section describes how your application can respond to Apple Events, and how to modify the event handling capabilities of the System Classes.

The Prograph Classic System Classes provide access to Macintosh events. The Macintosh Event Manager creates an event every time a user clicks the mouse, presses a key, or inserts a disk. Other events occur when your application's windows need to be redrawn and when your application is brought to the foreground or sent to the background.

For many types of applications you can use the System Classes as they are: all events will be handed automatically for you. However, by passing all events to your application, Prograph gives you the flexibility to modify the behavior of the System Classes to suit the needs of your application.

### Events in Prograph <sup>\*158\*</sup>

Events are passed to your application through a method, called Notify, in class Application. Every time an event occurs Notify is called, with three inputs: the current application, a Macintosh event record, and an integer which represents the type of event. For most events, Notify calls other methods to actually process the event. Classes Application, Window, Window Item, Canvas, Scroll List and Edit Text contain event handling methods. These methods use a set of Prograph primitives, called the System Class primitives, to aid in event handling.

### Apple Events <sup>\*158\*</sup>

Apple events are a new type of event for Macintosh System 7. They allow communication between applications on a single computer or across a network. For example, an Apple event is sent to your application when the user double-clicks on one of your application's documents.

There are four events called required events, which your application must respond to in order to be System 7 friendly. These are Open Application, Open Documents, Print Documents and Quit Application. <sup>\*158\*</sup>

Every Apple event is identified by a four character Event Class and a four character Event ID. The four required events, for example, all have the same class but different IDs: <sup>\*158\*</sup>

Apple Event	Class	ID
-------------	-------	----

Open Application		
'aevt'		
'oapp'		

Open Documents		
'aevt'		
'odoc'		

Print Documents	'aevt'	
'pdoc'		

## Quit Application

'aevt'

'quit'

The Event Class is used to group related Apple events into event suites. The four events with class 'aevt' make up the Required suite. Other suites include the Core suite, the Text suite, and the Graphical suite.

Besides class and ID, Apple events contain information about the sender of the event and other event-specific data. For example, the Open Document event contains a list of documents to be opened by your application. The Apple Event Manager has a collection of Macintosh methods for extracting information from Apple events.

In the Prograph System Classes, the attribute aevent methods of class Application is used to control Apple events which are received by your application. For each Apple event you wish to support, you specify the Event Class, the Event ID, and the method to be called when that event is received. The Application Editor contains an Apple Events button which opens the Apple Event Editor. <sup>1150</sup>

he Apple Event Editor includes three fields for entering an Event Class, Event ID, and the name of the method which is to respond to that combination. When the Insert button is pressed, these are then inserted into a scrolling list of such Apple Event Class/ID/Method triplets. A triplet in the scrolling list may be selected and then edited or deleted. <sup>1160</sup>

When your Prograph application receives an Apple event, the Notify method is called, just as it is for any other type of event. Three System Class primitives are used to dispatch Apple events: sc-aevent-begin, sc-aevent-dispatch and sc-aevent-end. sc-aevent-begin returns the Event Class and Event ID as well as the Apple event record and a default reply. The Apple event record contains all of the information associated with the Apple event. The Apple Event Manager automatically creates a default reply which is sent back to the sending application when you call sc-aevent-end. If you encounter an error while processing the Apple Event you can add error information to the default reply. sc-aevent-dispatch searches the aevent methods attribute and returns the method associated with the Event Class and Event ID. If it doesn't find a matching method, it returns NULL. <sup>1160</sup>

## Handling Events Yourself<sup>160</sup>

As mentioned earlier, events are passed to your application through the Notify method in class Application. Notify has three inputs: the current application, the event record, and the type of event. What follows is a brief description of events and event records; for more detailed information see Inside Macintosh.<sup>160</sup>

A Macintosh event record contains the event code, where and when the event occurred, and other information specific to each type of event. An event record contains the following fields:<sup>161</sup>

what	the event code (mouseDown, keyDown, etc.)
where	the location of the mouse when the event occurred, in global coordinates
when	the time the event occurred in ticks (sixtieths of a second) since system startup
message	depends on the event code
modifiers	depends on the event code

### The what field<sup>161</sup>

The what field contains an event code which identifies the type of event. This is passed as the third input to Notify for easy access. The event codes are available in Prograph as Macintosh constants. The most common events are:

nullEvent	No other event is pending.
mouseDown	The user pressed the mouse button.
keyDown	The user pressed a key on the keyboard.
autoKey	The user is holding down a key.
updateEvt	A window needs to be redrawn.
activateEvt	One of your application's windows has been selected (made the frontmost, active window), or deselected.
osEvt	Your application has been suspended (switched into the background), or resumed (brought into the foreground).
kHighLevelEvent	An Apple event has been sent to your application.

Other events which most applications will not need to handle are: keyUp, mouseUp, networkEvt, diskEvt, driverEvt, app1Evt, app2Evt, and app3Evt.

### The message field<sup>161</sup>

The event message contains information pertinent to keyDown, activate, update and suspend/resume events. For keyDown events the message contains, among other things, the ASCII code of the character that was typed. The code fragment below illustrates how to extract the character from the event message

of a keyDown event.

or activate and update events the event message contains a pointer to the Macintosh window record of the affected window. Use the primitive `sc-get-wind` to find the Prograph window affected by update and activate events.

When the event code is `osEvt`, the event message indicates whether your application is being suspended or resumed. If the message is an even number (bit 0 is off) your application is being suspended; if the message is an odd number (bit 0 is on) your application is being resumed. The figure below illustrates how to test the event message of an `osEvt` event.

he modifiers field

The event modifiers apply to key and activate events. For activate events, bit 0 indicates whether the affected window is being activated or deactivated. If bit 0 is on, the window is being activated; if bit 0 is off, the window is being deactivated. The code below illustrates how to test the event modifiers of an activate event.

or key events the modifiers indicate the state of the modifier keys: Command, option, shift, control, and caps lock. The figure below illustrates how to determine the state of the modifier keys in Prograph.<sup>163</sup>

mdKey is a Macintosh constant used to extract the state of the command key. Use the constants shiftKey, alphaLock, optionKey and controlKey for the other modifier keys.<sup>163</sup>