Using the Notification Manager

To issue a notification to the user, you need to create a notification request and install it into the notification queue. The **Notification Manager** interprets the request and presents the notification to the user at the earliest possible time. After you have notified the user in the desired manner (that is, placed a diamond mark in the Application menu, added a small icon to the list of icons that alternate in the menu bar, played a sound, or presented the user with an alert box), you might want the **Notification Manager** to call a response procedure. The response procedure is useful for determining that the user has indeed seen the notification or for reacting to the successful posting of the notification. Eventually, you will need to remove the notification request from the notification queue; you can do this in the response procedure or when your application returns to the foreground.

The **Notification Manager** is automatically initialized at system startup time. It includes two functions, one that allows you to install a request into the notification queue and one that allows you to remove a request from that queue.

Creating a Notification Request

Information describing each notification request is contained in the notification queue, which is a standard Macintosh queue, as described in the **Operating System Utilities**. When installing a request into the notification queue, your application must supply a pointer to a notification record that indicates the type of notification you desire. Each entry in the notification queue is a notification record-a static and nonrelocatable record of type **NMRec**.

```
 // How to set up a notification record.

     NMRec myNotification;         //a notification record
     short           myResNum;           //resource ID of small icon resource
     Handle  myResHand;            //handle to small icon resource
     Str255          myText = "\pSample Alert Box";
                                         //string to print in alert box

     myResNum = 1234;                    //resource ID in resource fork
     myResHand = GetResource('SICN', myResNum);
                                         //get small icon from resource fork

     myNotification.qType = nmType;      //set queue type
     myNotification.nmMark = 1; //put mark in Application menu
     myNotification.nmIcon = myResHand;//alternating icon
     myNotification.nmSound = (Handle)-1;      //play system alert
                                               // sound
     myNotification.nmStr = myText;            //display alert box
     myNotification.nmResp = nil;              //no response procedure
     myNotification.nmRefCon = 0;                    //not needed
```

This notification record requests all three types of notification-polite (alternating small icon), audible (system alert sound), and alert (alert box). In addition, the diamond appears in front of the application's name in the Application menu. In this case, the small icon has resource ID 1234 of type

'SICN' in the application's resource fork. The nmIcon field could also have had an icon family containing a small icon assigned to it. See **Drawing Icons with System 7** for more on icon families.

Defining a Response Procedure

The nmResp field of the notification record contains the address of a response procedure that executes as the final stage of a notification. If you do not need to do any processing in response to the notification, then you can supply the value NULIL in that field. If you supply the address of your own response procedure in the nmResp field, the **Notification Manager** passes it one parameter, a pointer to your notification record. For example, this is how you would declare a response procedure having the name **MyResponse**:

```
pascal void MyResponse (NMRecPtr nmReqPtr);
```

When the **Notification Manager** calls this response procedure, it does not set up **A5** or low-memory global variables for you. If you need to access your application's global variables, you should save its **A5** in the nmRefCon field. See the **Memory Manager** and Apple Technical Note #180 for more information on saving and restoring the **A5 world**.

Response procedures should never draw on the screen or otherwise affect the human interface. Rather, you should use them simply to remove notification requests from the notification queue and free any memory. If you specify the special nmResp value of -1, the **Notification Manager** removes the queue element from the queue automatically, so you don't have to do it yourself. You have to pass your own response routine, however, if you need to do anything else in the response procedure, such as free the memory block containing the queue element or set an application global variable that indicates that the notification was received.

If you choose to use audible or alert notifications, you should probably use an nmResp value of -1 so that the notification record is removed from the queue as soon as the sound has finished or the user has dismissed the alert box. However, if either nmMark or nmIcon is nonzero, you should not use an nmResp value of -1 because the **Notification Manager** would remove the diamond mark or the small icon before the user could see it. Note that an nmResp value of -1 does not free the memory block containing the queue element; it merely removes that element from the notification queue.

Since the response procedure executes as the last step in the notification process, your application can determine that the notification was posted by examining a global variable that you set in the response procedure. In addition, to determine that the user has actually received the notification, you need to request an alert notification. This is because the response procedure executes only after the user has clicked the **OK** button in the alert box.

Installing a Notification Request

The **Notification Manager** includes two functions, one to install a notification request and one to remove a notification request. To install a notification request, use the function **NMInstall**.

To add a notification request to the notification queue, call **NMInstall**. For example, you can install the notification request defined in the Listing above with the following line of code:

```
myErr = NMInstall (&myNotification);      install notification request
```

Before calling **NMInstall**, you should check to make sure that your application is running in the background. If your application is in the foreground, you do not need to use the **Notification Manager** to notify the user; instead, you can simply use standard methods for playing sounds or putting up alert boxes.

> **Note:** VBL tasks, **Time Manager** tasks, and device drivers that want to install notification requests do not need to make this check because they are never in the foreground. Generally, however, a VBL task or a **Time Manager** task can avoid issuing notification requests by setting a global flag that informs the application that installed it that a notification needs to be requested. When that application receives some processing time, it can alert the user in the appropriate manner (that is, by putting up an alert box or by issuing a notification request). This method allows you to keep interrupt-time tasks, such as VBL and **Time Manager** tasks, small and quick.

If the call to **NMInstall** returns an error, then you cannot install the notification request in the notification queue. In that case, your application must wait for the user to switch it into the foreground before doing further processing. While waiting for a resume event, your application should take care of other events, such as updates. Note, however, that the only reason that **NMInstall** might fail is if it is passed invalid information, namely, the wrong value for qType.

You can install notification requests at any time, even when the system is executing 'INIT' resources as part of the system startup sequence. If you need to notify the user of some important occurrence during the execution of your 'INIT' resource, you should use the **Notification Manager** to install a request in the notification queue. The system notifies the user after the startup process completes, that is, when the normal event mechanism begins. This saves you from having to interrupt the system startup sequence with dialog or alert boxes and results in a cleaner and more uniform startup appearance.

NMInstall has a single parameter, nmRecPtr, which is a pointer to a notification record. It adds the notification request specified by that record to the notification queue and returns a result code.

> **Note:** **NMInstall** does not move or purge memory, so you can call it from completion routines or interrupt handlers as well as from the main body of an application and from the response procedure of a notification request.

Removing a Notification Request

To remove a notification request from the notification queue, call **NMRemove**. For example, you can remove a notification request with this code:

```
myErr = NMRemove (&myNotification);     remove notification request
```

You can remove requests at any time, either before or after the notification actually occurs. Note that requests that have already been issued by the **Notification Manager** are not automatically removed from the queue.

NMRemove removes the notification request identified by *nmReqPtr* from the notification queue and returns a result code.