New Technical Notes

Macintosh

Developer Support

Notification Manager Q&As

Processes **M.PS.NotifMgr.Q&As**

Revised by: Developer Support Center October 1992
Written by: Developer Support Center October 1990

This Technical Note contains a collection of Q&As relating to a specific topic—questions you've sent the Developer Support Center (DSC) along with answers from the DSC engineers. While DSC engineers have checked the Q&A content for accuracy, the Q&A Technical Notes don't have the editing and organization of other Technical Notes. The Q&A function is to get new technical information and updates to you quickly, saving the polish for when the information migrates into reference manuals.

Q&As are now included with Technical Notes to make access to technical updates easier for you. If you have comments or suggestions about Q&A content or distribution, please let us know by sending an AppleLink to DEVFEEDBACK. Apple Partners may send technical questions about Q&A content to DEVSUPPORT for resolution.

New Q&As and Q&As revised this month are marked with a bar in the side margin.

Macintosh background DA alerts
Written: 6/10/91
Last reviewed: 8/1/92

How can I use the Notification Manager to alert the user to bring my DA to the front and then display an alert?

—

Basically, if your DA is in the foreground and you finish your task, you can just use Alert() to notify the user. If your DA is in the background when it finishes, post a notification with NMInstall that notifies the user by sound, icon, and mark in the process menu (but not an alert) that something is up in your desk accessory. When they switch to your DA, you then display your alert.

The following code is for a DA written in Think C that notifies the user as described above. Portions of the code are borrowed from the Think C example "Hex Dump DA." Basically, this code keeps track of whether the DA is in the foreground or background by setting the global variable gInBackground at every activate and deactivate event. If there is a

notification pending and an activate event is received, the routine MyResponse is called to remove the notification and display the appropriate alert.

Note that for MPW you'll have to store your globals elsewhere, as MPW doesn't provide the Think C A4 feature. You'll have to modify the code to work with your function to get the proper resource IDs.

```
void PostNotify(StringPtr theStr)
{
   extern Boolean gInBackground;    /* true=in background */
   extern NMRec *gNotify;   /* global var pointing to notif. rec 0=none */
   Handle notIcon;
   Ptr txtPtr;

   /* blast away any other pending notification */

   if (gNotify) {
       DisposPtr((Ptr) gNotify->nmRefCon);
       NMRemove(gNotify);
       gNotify = nil;
   }

   /* see if we should notify or use alert */

   if (gInBackground) {

       /* allocate memory for notification */

       txtPtr = NewPtr(sizeof(Str255));
       if (MemError()!=noErr)
           return;
       gNotify = (NMRec *)NewPtr(sizeof(NMRec));
       if (MemError()!=noErr)
           return;

       /* copy notif. string to dynamically allocated block */

       BlockMove(theStr,txtPtr,256);

       /* fill out notif. record */

       notIcon = GetResource('SICN',OwnedResourceID(NOTIF_ICON));
       gNotify->nmStr = nil;
       gNotify->qType = nmType;
       gNotify->nmMark = 1;
       gNotify->nmIcon = notIcon;
       gNotify->nmSound = (Handle)-1;
       gNotify->nmResp = nil;
       gNotify->nmRefCon = (long)txtPtr;

       NMInstall(gNotify);
   }
   else {
       ParamText(theStr,"\p","\p","\p");
       StopAlert(OwnedResourceID(ALERT_DIALOG),nil);
   }
}

/* response procedure-- called on activateEvt */

pascal void MyResponse(QElemPtr nmReqPtr)
{
   StringPtr errTxt;

   errTxt = (StringPtr) ((NMRec *)nmReqPtr)->nmRefCon;
   NMRemove((NMRec *)nmReqPtr);

   ParamText(errTxt,"\p","\p","\p");
   StopAlert(OwnedResourceID(ALERT_DIALOG),nil);

   DisposPtr(errTxt);
```

```
}

/*-----------------------------------------------------------*/
 here's the piece of the code that handles activate events
/*-----------------------------------------------------------*/

doActivate(activate)
Boolean activate;
{
   extern Boolean gInBackground;
   extern NMRec *gNotify;
   Rect r;
   GrafPtr    savePort;

   GetPort(&savePort);
   SetPort(wp);
   r = wp->portRect;
   r.top = r.bottom - 16;
   r.left = r.left - 16;
   InvalRect(&r);
   gInBackground = !activate;
   if ( activate ) {
       if (gNotify) {
           MyResponse((QElemPtr)gNotify);
           gNotify = nil;
       }
   }
   SetPort(savePort);
}
```

Valid resource handles for Notification Manager nmIcon field

Written: 6/3/91
Last reviewed: 8/1/92

Why can’t I pass a handle to an 'ICN#' as the Notification Manager nmIcon under System 7?

—

The Notification Manager chapter of Inside Macintosh Volume VI describes the nmIcon field as follows:

nmIcon: Contains a handle to a small icon that is to appear periodically in the menu bar...

The nmIcon field holds a handle to the icon that flashes in the process menu. Since 'ICN#' are not small (16x16), they will not work for this purpose. You need to pass a 'SICN' handle in the parameter block. Changing the icon that appears in the notification alert dialog box is not supported.