Tips for Font Handling in Macintosh Applications

Adobe Developer Support

Technical Note #5137

23 September 1998
Tips for Font Handling in Macintosh Applications

1 Introduction

For applications, font handling includes managing the outline fonts (including font names and ID numbers) available on the system, font resources to keep track of activation and deactivation of fonts, and the font menu and the potentially numerous updates of it while the application is open. This document explains how the Adobe® Type Manager® (ATM®) software can be used by an application to aid in font management, particularly for situations where documents reference fonts that are not installed in the user’s system. Instead of the user getting an error message, ATM can auto-activate an uninstalled font, generate missing instances of a multiple master fonts, or perform font substitution in a manner that preserves the look of the original document. In addition, this document explains how to manage fonts so that the user’s font menus are updated when either the user installs or removes fonts, or when ATM auto-activates or deactivates fonts during an application session.

Most of the features referenced in this document are available to users who have ATM 4.0 Deluxe installed in their system. The version of ATM that is bundled with applications and earlier versions of Adobe Acrobat®, does not support those features.

2 ATM and Missing Font Problems

A major frustration can occur when users open a document that references fonts that are not installed in their system. ATM 4.0 provides a solution for this problem for the following three cases:

- The font is in the user’s system, but it is not currently installed. If the ATM software can find the font in the user’s system, it will activate the font automatically.

- The font is not in the user’s system, but its name is in the ATM font database. ATM will create a simulation font—a process that creates a substitute font that matches the general style (serif or sans-serif), weight, slant, x-height, and character widths of the original font. This means that
the document will retain the look and feel of the original, as well as retaining the correct line and page breaks. The ATM font database contains metrics for most fonts in the Adobe Type Library®, as well as for a number of core TrueType fonts from the Macintosh.

- The font is an installed multiple master font, but the specific instance referenced in the document does not exist. ATM will create the appropriate instance and install it in the user’s system.

You can correctly handle these cases by using the Macintosh system call `GetFNum`, and by tracking fonts by the font name, and not by their font ID numbers.

3 Application Compatibility Guidelines

The following guidelines describe how your application can use ATM to do a better job of managing fonts.

Identifying Fonts by Name

First and foremost, remember that fonts are identified by name in documents, not by font ID. An application can get the font ID, given the font name, and assume that the font ID will not change while the application is still running. However, after the application quits, any suitcase’s font ID (except the system standard Chicago, Monaco and Geneva) may change before the application is relaunched.

Use GetFNum for all Referenced Fonts not in the Resource Chain

You should not assume that because a font is not currently in the resource chain, that it is unavailable. It is only when the application asks for a font that is not currently in the resource chain, that ATM can know that a font is not available. It can then do its job of trying to either activate the font, create a substitute for the font, or create an instance of a multiple master font.

The lowest cost and safest Macintosh system call to find out if a font can be made to exist is `GetFNum()`. The lowest cost ATM API call for the same function is `IsFontAvailableATM()`. `GetFNum()` is called with a font name, and the system returns a font ID number that can be used with QuickDraw functions such as `TextFont()`.

Rebuild Font Menus Periodically

An application should periodically check to see if the font resource chain has changed, and if so, rebuild the font menus. There is a single resource, type ‘SCoc’ id 0, available to all applications when ATM (or Suitcase II or 3) is running. It’s a handle that points to a 16-bit unsigned integer. Every time any suitcases are activated or deactivated, the integer gets incremented. When the
application starts up, the value of that integer should be saved. Anytime that
value changes, the application should rebuild its font menus (using
`AddResMenu()` and any other structures relating to the fonts it is managing.
For example, applications that maintain a parallel array of font IDs and font
menu entries should update both.

The application should at least check SCoc’s value:

• Every time it comes to the foreground from the background

• Before showing the font menu at the user’s request

• Just after it reads in a new document that may contain new fonts

Note the “after”—asking if fonts exist will cause them to be added to the
resource chain (assuming that ATM is successful). The list of fonts available
after a document has opened may be quite a bit longer than before. The
following describe the ways in which the font list can change:

• The user switches to the ATM application window and manually activates
  or deactivates fonts

• The user activates and deactivates fonts from the font menu with Adobe
  Type Reunion® (ATR) software

• The ATM software automatically activates or substitutes a font when it is
  requested in a document.

Following the above guidelines for checking SCoc’s value will keep an
application’s Fonts menu current with the fonts that ATM has activated and
deactivated. Another safety check is when the Fonts menu is picked, instead
of using a parallel array of font IDs, get the menu text item and call
`GetFNum()` to get the font ID. This will avoid any problems with fonts that
have been deactivated, and will work with Adobe Type Reunion 2.0 when it
lists fonts that have not actually been activated.

**Use AddResMenu() to build all font menus**

This will make sure that Adobe Type Reunion will work with your
application’s font menus. It will provide your application with a healthy
speed boost: ATR caches the standard font menu entries, and can speed
opening an application by more than 2x when only 40 fonts are loaded. It is
also much faster than the optimized `AddResMenu()` in Mac OS 7.5.5.
Don’t Keep Pointers into Font Resources

Applications should not keep pointers into FOND resources, or any other font resources. Deactivation may make those resources obsolete and cause the application to crash. Make sure that the code does not depend on handles remaining valid between calls to *WaitNextEvent*.

4 Storing and Using Font Names

The following guidelines for handling font names will help to ensure more seamless support for fonts:

- Do not store font ID numbers in your document, store font names.

- Keep the original font names in the document, even if they cannot be found in the system.

- When opening a document and a particular font is not found, call `GetFNum()` with the font name. If the name is a multiple master instance, ATM will create the instance. If a font’s name is listed in the ATM database, ATM will create and display a simulation font with matching widths and a similar look, to be used while the document is open.

- Make `GetFNum()` calls only for the current document when the fonts menu is selected by the user. It is unnecessary to use `GetFNum()` on all the fonts in the system for such a request.

Adobe provides an application programming interface (API) for ATM which provides valuable functionality for font handling. A few of the ATM API calls are described here, but all calls are fully documented in Adobe Technical Note #5072, *Adobe Type Manager Software API: Macintosh*.

Some applications store the PostScript® language font name instead of Macintosh menu names (see Technical Note #5088, *Font Naming Issues* for more information). When they can’t find a font, they can call the ATM API function `getPSNumATM()` with the PostScript font name. If the name is in the ATM font database, ATM will simulate the font and return its font ID number.

When the function `getPSNumATM()` is called, ATM does not check if the real font is installed in the system; instead it always creates the simulation font if the appropriate data exists in its database. Because of this, you should first check whether the required font is already installed on the system. The Adobe Developers Association has C language source code available for a function called `getFontFace()` that, given a PostScript language font name, returns the corresponding Macintosh font ID and style if the font is installed in the
system. If the Macintosh font menu name is desired, the font ID returned by
\texttt{getFontFace()} can be passed to the toolbox call \texttt{GetFontName()} which
returns the Macintosh menu name.

Before placing fonts in the font menu, you should call the ATM API function
\texttt{isSubstFontATM()} to check if the font is a simulation font. If it is, append a
Chicago 12-point solid black diamond (character code $13$) to the font name
to indicate that it is a font simulated by ATM.

If you already have a Macintosh font name and wish to see if the original font
is available, call the function \texttt{GetNamedResource(‘FOND’, fontName)}. If a
non-null handle is returned, pass \texttt{fontName} to the ATM API function
\texttt{isSubstFontATM()} to see if it is a previously created font. If \texttt{GetFNum()} is
called for this check, it will always indicate that the font is found since ATM
has created a simulated font.

4.1 Updating Font Menus “On the Fly”

Since fonts can be added to the system at any time during the execution of an
application—for example they might be added by the user creating a multiple
master font instance, or a new suitcase might be added using a font manager
utility. Hence it is essential to make certain that font menus are kept up-to-
date. The following information will assist in determining when to update
your font menus and tables.

Version 2.0 and higher of Suitcase™ II creates a resource (‘SCoc’ – SuiteCase
Open Count) that is incremented each time a suitcase is opened or closed.
ATM also increments ‘SCoc’ whenever ATM creates a font, or whenever a
font is dragged into the system folder

Additionally, ATM will create ‘SCoc’ if Suitcase II is not available.

\textbf{Note} Applications should never do a \texttt{ReleaseResource()} on the ‘SCoc’ resource
since it is a shared resource located in the system heap. Doing so will cause
problems for both ATM and Suitcase. There is no need to do any special
cleanup of this resource before your application quits.

Applications should check ‘SCoc’ often to see if the font menu needs
updating. The following sample code checks the ‘SCoc’ resource:

\begin{verbatim}
static unsigned short **SuitcaseCount;
static unsigned short lastCount;

/* Call InitSuitcaseCount( ) when the application
 is first initialized.*/

void InitSuitcaseCount(void)
{
    SuitcaseCount = (unsigned short **) GetResource('SCoc',0);
\end{verbatim}
if (SuitcaseCount != NULL)
    lastCount = **SuitcaseCount;
}

/* Call CheckSuitcaseCount( ) fairly frequently.
 * In an idle loop is fine (this is VERY fast if nothing
 * has changed!).
 */

void CheckSuitcaseCount(void)
{
    if (SuitcaseCount != NULL)
        if (**SuitcaseCount != lastCount)
            {
            /***** UPDATE APPLICATION MENUS & TABLES HERE *****/
            lastCount = **SuitcaseCount;
            }
}