
from Hands High Software, Inc.

# Developer's Guide

## Introduction

FontBucket is a Palm application that lets users manage a central repository of fonts. Users can install fonts using the Palm Install Tool, delete them, and even beam them.

FontBucket also allows Palm programmers to use this group of fonts in their own programs. No longer will programmers have to define their own fonts, separate from the fonts of all other programs. FontBucket gives users and programmers a way to manage fonts from a single location just like on the desktop.

## How Much Is It?

FontBucket is FREE. You may freely distribute it with your own software.

## How Does It Work?

FontBucket's API makes calls into FontBucket using launch codes. FontBucket does not do any trap patching or use hacks. It works in ways entirely supported by the Palm OS, and so should be able to work on future ARM processors and other Palm compatible platforms.

FontBucket includes a .c and .h file that gives you a single API to access fonts. If for some reason FontBucket is not installed on the user's machine, the FontBucket API will automatically make calls to standard PalmOS calls instead. You do not need to worry about checking for the existence of FontBucket.

## Using the FontBucket API

The general idea is as follows.

In the initialization code for your program, call FmInit to initialize font bucket. This will allocate memory that font bucket needs to map Palm fonts with fonts in its database.

Normally, when working with fonts, programmers use an 8-bit FontID. FontBucket's API defines a 32-bit FmFontID that is a uniqueID for a font. FontBucket maps FontIDs to FmFont Ids throught the FmUseFont function. Between launches, the FontID of a font might change, but the FmFontId of a particular font will not change. Therefore, you should save the FmFontId of a font in a preference file, instead of the FontID.

To bring up a font selection dialog box, allowing the user to choose from all installed fonts, call FmSelectFont. This will return an FmFontID. This FmFontID is a unique id associated with the font, and you should save and restore the FmFontID into your preference file.

Before you draw with a font, call FmUseFont to get a Palm FontID to draw with. FmUseFont associates your FmFontID with a Palm FontID in an internal lookup table. You then call FntSetFont with the FontID returned from FmUseFont.
When you are finished with a font, call FmFreeFont to free up space in the font table for other fonts the user might select. You can have at most around 125 active fonts in use at any one time. In other words, you can have 125 fonts that you have associated with FontIDs using FmUseFont, and not released using FmFreeFont.

One thing to watch out for is that you should not free a font that is in use. In particular, if you have set a field or other form object to a font, and you then call FmFreeFont on that font, the Palm will crash, even if you never draw that object again. The reason is that eventually your form will close, which will cause the form objects in the form to get deleted. During this delete process, the Palm OS validates the internal data in the form objects. If the font it is using has been deallocated, or changed, it will crash. No, the validation code won't just fail, it will crash.

For example, a typical thing you might do is as follows:

```
FmFontID  fmFontID = 0;
FontID         fontID = stdFont.
          oldFont;

FmSelectFont (fmPtr, &fmFontID);

FmUseFont (fmPtr, fmFontID, &fontID);

oldFont = FldGetFont (fld);

FldSetFont (fld, fontID);
```

```
FldDrawField (fld);

FmFreeFont (fmPtr, oldFont);
```

Note that FmFreeFont is called after FldSetFont.

When your application exits, call FmFreeFont on any current fonts in use, then FmClose to deallocate memory used for the font table. If you did not call FmFreeFont on every font in use, FmClose will return an error code.

## Incorporating FontBucket into Your Project

Include the FontMgr.c and FontMgr.r files into you project window or build list, and have the FontMgr.h file in the directory path of your include files. When you are ready to run, make sure you install the FontBucket.prc file and any font files you want as well.

If you are having trouble adding the FontBucket.r file, make sure that your File Mapppings setting in CodeWarrior is set up correctly. Recent versions of CodeWarrior do not have .r files set up by default. To set up .r files, do the following:

1) Choose Settings from the Edit menu
2) Tap on the File Mappings icon
3) Enter nothing in the File Type box.
4) Enter ".r" in the Extension box
5) Choose Rez from the Compiler menu
6) Tap the Add button

Here is an example of the changes we made in ToDo PLUS to support FontBucket. ToDo PLUS already supported the Palm standard fonts.

In ToDoPlus.h, the preference structure now includes an fmFontID as in:

```
typedef struct
{
     …
     FmFontID    fmListFontID;
} ToDoPlusPreferenceType;
```

In StartApplication():

```
FmInit(&FontBucketParams, kNoFontRangeSpecified,
kNoFontRangeSpecified);
```

```
if (prefsVersion == noPreferenceFound) {
     FmGetFMFontID(&FontBucketParams, stdFont,
&fmFontID);
     FmUseFont(&FontBucketParams, fmFontID, &ListFont);
} else {
FmUseFont(&FontBucketParams, prefs.fmListFontID,
&ListFont);
}
```

In StopApplication():

```
FmGetFMFontID(&fmType, ListFont, &fmFontID);
prefs.fmListFontID = fmFontID;
```

…

```
FmFreeFont (&FontBucketParams, ListFont);
FmClose(&FontBucketParams);
```

In the ToDo PLUS Form, in DoMenuCommand():

```
case ViewOptionsFont:
{
    FontID   previousFontID = ListFont;
    FmFontID fmFontID;
    Boolean  fontChanged;

     ListViewClearEditState ();

     FmGetFMFontID(&FontBucketParams, ListFont,
&fmFontID);
     fontChanged = FmSelectFont(&FontBucketParams,
&fmFontID);

     if (fontChanged){
          FmUseFont(&FontBucketParams, fmFontID,
&ListFont);
          FmFreeFont(&FontBucketParams, previousFontID);
          FrmUpdateForm (ViewForm, updateCompleteRefresh);
     }
     handled = true;
}
break;
```

```
That is all!
```

## Handera Support

FontBucket supports the Handera screen and the additional Handera fonts. To take advantage of the Handera hi-res screen, you must use the Handera API to put your application into hi-res mode. If you do not put the screen in hi-res mode, the Handera will use an algorithm to increase the size of the letters 50%, and the results will not be that nice. However, in hi-res mode, all fonts appear smaller. The user will have to choose a larger font to have the font appear the same size as on a standard screen.

The Handera includes some extra built-in fonts. Be aware that the Handera uses 0xE0 and above as the font ids for its VGA fonts, so you cannot use that range for your software. That also reduces the number of fonts that you can simultaneously define with FontBucket, but you can still define over 90 fonts at one time.

If your application includes support for the Handera, you should define HANDERA_SUPPORT in your precompiled header. This will enable FontBucket to provide the user with the additional Handera fonts in the font selection screen.

## Sony Support

FontBucket does not currently support hi-resolution in the Sony per-se, but you can still use FontBucket to draw hi-res fonts. Confused? Well, here is the deal.

Sony does not fully support custom fonts. If you put the Sony device in hi-res mode, you cannot use a custom font in a Palm form object, like a field or button, nor can you draw directly to the screen. If you do, you will get garbage on the screen, and it might crash. However, what you CAN do is use a custom font and draw into an offscreen buffer that you then copy to the main window. You must use HRFntSetFont and HRWinDrawChars to do so. When you do this, the fonts will appear 50% smaller than on a regular screen. We know of some applications that are using this technique with FontBucket on a Sony device.

Sony has publicly stated that they intend to support the hi-res font format of OS 5, and we intend to support that as well. This font format will allow much more flexibility when drawing onto hi-res screens. So, OS 5 is the planned solution to this problem.

## Where do I get fonts?

AlphaFonts and CoolFonts are two popular font collections that are already included with the program. More fonts will be downloadable from our Web site as they become available at http://www.handshigh.com/fontbucket/.

## Making Your Own Font Collections

FontBucket defines a new font database format that allows you to include multiple font records in each database, giving each font a name, style and size. The FontBucket import file format is described below.

FontBucket import files have the file type of “xFnt” and a creator of “FMGR”. FontBucket supports the FontHack database format as well, but the FontBucket database format is recommended.

Included with the Macintosh version of FontBucket is a utility called FBConvert. If you drag a Macintosh font suitcase file onto FBConvert, it will create a FontBucket import file using the bitmapped fonts in that suitcase file. See the FBConvert ReadMe for more information.

Also, on Apple’s Web site, there is a program called Fissioner which will take a TrueType font and convert it to a bitmap font. This can be used in conjunction with FBConvert to convert a TrueType font into a FontBucket font.

To get Fissioner, do the following:
1) Go to http://developer.apple.com/fonts/Tools/
2) Click the "font tools license" link in the first paragraph
3) Click the Accept button in the frame at the bottom
4) Scroll down the page until you see Fissioner. Click the link to download.

When using Fissioner, be sure to select “Installable Font”, instead of “Fussioner Input” when going through the dialogs.

Another way is to use FontHack files, as follows:

1) Create fonts in the FontHack format and install them in the device (or Palm Emulator).
2) Start FontBucket
3) Click Import when FontBucket detects the FontHack files
4) Edit the names and styles of the fonts as appropriate
5) If you are using the Emulator, exit FontBucket and export the FontBucketDB file to your desktop computer. If you are using a real device, HotSync, and look for the FontBucektDB.pdb file and make a copy of it.
6) Use a hex editor to change the database name inside the .pdb file to something else, and change the type code from “DATA” to “xFnt”. The database name is the first thing you will see in the file, and the type is at hex offset 0x3C.

This file can now be installed into your Palm device.

Please follow these conventions for style codes:
B – Bold

I – Italic
N – Narrow, Condensed
X – Wide, Expanded
L – Light

## The FontBucket Database Format

Both FontBucket import files, and the FontBucket database have the same format. The only difference is that the FontBucket database is named “FontBucketDB” and has a type of “DATA”, while import files can be named anything else and have a type of “xFnt” and a creator of “FMGR”.

Each record of the database contains information on one font. Each record is formatted as follows:

```
typedef struct{
        UInt8           version;            // version of data format, currently 0
        UInt8           systemFont;         // if this is not zero, then this structure has no
        font data, and instead
                                            // refers to a system font. And it with 0x7f to get
the system font value.
        UInt16          size;           // Pixel height of font. For display purposes only
        UInt16          reserved;       // For future expansion, will likely contain flags to
specify different kinds of font data.
        UInt32          fontDataSize;   // The size in bytes of the font data

        // font data is embedded immediately after the fontDataSize, and before the variable
length strings. This is the NFNT format of the Macintosh, with an extra blank pixel on the right.
                                                                    .
        Char            fontName;   // variable length - zero terminated name of the font
        Char            style;      // variable length - zero terminated style name
} EmptyFontRecordType, *EmptyFontRecordPtr;
```