Technical Note TN2012

Building QuickTime Components for Mac OS X

CONTENTS

This Technote discusses building CFM and Mach-O QuickTime components for Mach OS X and illustrates the changes required to facilitate moving existing QuickTime components to Mac OS X.

Updated: [Mar 8 2000]

Binary Formats

Mac OS X supports two application binary formats; CFM and Mach-O. CFM (Code Fragment Manager) is used on Power Macintosh Computers running traditional Mac OS today and is supported by compilers such as Metrowerks CodeWarrior. Mach-O is the native Mac OS X object format and is supported by gcc.

Carbon components for Mac OS X may be built as CFM or Mach-O code.

Back to top

Carbon CFM Components

CFM components for Mac OS X are built using similar mechanisms to those for Mac OS 7, 8 and 9 with the following differences:

a. Carbon CFM components must link to CarbonLib instead of InterfaceLib, QuickTimeLib, etc.
b. The code must reside in the data fork of the file, as Mac OS X does not support component code in resources. The resource fork must contain an extended 'cfrg' resource with a section referencing the code. See Listing 1.
c. The "code resource type" entry in the 'thng' resource should be 'cfrg'. The "code resource ID" entry should not be a real resource ID. If there is only one member in the 'cfrg' resource, then the "code resource ID" can be zero. If there are several, then the "code resource ID" should contain a value that matches an extension field (qualifier 1) in the appropriate 'cfrg' member. See Listing 2.
d. The main entry point of the code should be the component dispatcher routine instead of a routine descriptor. There are no routine descriptors in Carbon on Mac OS X. On Mac OS X a UPP is not a pointer to a routine descriptor.
e. The platform type should be platformPowerPCNativeEntryPoint, which is defined as the value 5, not platformPowerPC which is defined as the value 2. See Listing 2.

Back to top

Merging multiple CFM Components into a single file

The 'cfrg' resource must have an ID of 0 and there can only be one 'cfrg' resource per file. The trailing part of a 'cfrg' resource is an array of entries used to associate a name and architecture type to a specified code fragment.

The CodeWarrior MacOS Merge Linker can be used to create a single file binary containing multiple Carbon CFM components for Mac OS X. This linker will concatenate each code fragment and will use the custom extended code fragment 'cfrg' (0) entry, from each component, to automatically build the array of entries. The merged 'cfrg' resource will contain the correct code fragment locations, offsets and length values.

MacOS Merge can be chosen from the CodeWarrior 'Target Settings' preference panel. Be sure to select the 'Copy Code Fragments' and 'Copy Resources' options from the MacOS Merge 'Linker Settings' preference panel.

For more information regarding 'cfrg' resources consult the CodeFragments.r header file and the Code Fragment Manager chapter of Inside Macintosh PowerPC System Software.

Listing 1 - CFM Custom `'cfrg'` resource

```
// Extended versions of the 'cfrg' resource template.
#define cfrg_RezTemplateVersion 1

#include <CodeFragments.r>

// Custom extended code fragment resource
// CodeWarrior will correctly adjust the offset and length of each
// code fragment when building a MacOS Merge target
resource 'cfrg' (0) {
 {
  extendedEntry {
        kPowerPCCFragArch,              // archType
        kIsCompleteCFrag,               // updateLevel
        kNoVersionNum,                  // currentVersion
        kNoVersionNum,                  // oldDefVersion
        kDefaultStackSize,              // appStackSize
        kNoAppSubFolder,                // appSubFolderID
        kImportLibraryCFrag,            // usage
        kDataForkCFragLocator,          // where
        kZeroOffset,                    // offset
        kCFragGoesToEOF,                // length
        "MyComponent",                  // member name

        // Start of extended info

        'cpnt',             // libKind (not kFragComponentMgrComponent == 'comp'
                        // as you might expect)
        "\0x01\0x00",       // qualifier 1 - hex 0x0100 (256) matches Code ID in 'thng'
        "",                 // qualifier 2
        "",                 // qualifier 3
        "My CFM Component", // intlName, localized
  };
 };
};
```

Listing 2 - CFM 'thng' resource

```
// extended 'thng' template
#define thng_RezTemplateVersion 1

#include <Components.r>

resource 'thng' (256) {
        kAQTComponentType,      // Type
        'DEMO',                 // SubType
        'demo',                 // Manufacturer
        0,                      // not used - use componentHasMultiplePlatforms
        0,
        0,
        0,
        'STR ',                 // Name Type
        128,                    // Name ID
        'STR ',                 // Info Type
        129,                    // Info ID
        0,                      // Icon Type
        0,                      // Icon ID
        kMyComponentVersion,    // Version
        componentHasMultiplePlatforms +     // Registration Flags
        myComponentRegistrationFlags,
        0,                                  // Resource ID of Icon Family
        {
          kMyComponentFlags,                // Component Flags
          'cfrg',                           // Special Case: data-fork based code fragment
          /* Code ID usage for CFM components:
            0 (kCFragResourceID) - This means the first member in the code fragment;
          Should only be used when building a single carbon component per file. When
          doing so using kCFragResourceID simplifies things because a custom 'cfrg'
          resource is not required
          n - This value must match the special 'cpnt' extension field (qualifier 1) in
            the custom 'cfrg' resource
        */
          256,                                  // Custom code ID
          platformPowerPCNativeEntryPoint,      // Platform Type (response from
                                          // gestaltComponentPlatform or failing that,
                                          // gestaltSysArchitecture)
        };
};
```

Back to top

Mach-O Components

Mach-O components for Mac OS X contain a dynamic library (dylib) in their data fork and are built using similar mechanisms to those for Mac OS 7, 8 and 9 with the following differences:

a. The entry point is found by symbol name using the same mechanism as on Windows -- the "code resource type" is 'dlle' and the 'dlle' resource contains a C string which is the exported symbol name. See Listing 3.
b. The platform type should be platformPowerPCNativeEntryPoint and not platformPowerPC. This is the same as Carbon CFM components. See Listing 4.

Project Builder will allow you to build both single file and bundled components for Mac OS X.

To build bundled components select 'Carbon Bundle' from Project Builders 'New Project' dialog, or 'Bundle' from the 'New Target' dialog if you're adding a target to an existing project.

To build single file components select 'Empty Project' from Project Builders 'New Project' dialog then add a 'Library' target. Be sure to change the targets build setting 'LIBRARY_STYLE' to 'DYNAMIC'.

Listing 3 - Mach-O Entry Point

```
// Code Entry Point for Mach-O and Windows
resource 'dlle' (512) {
        "MyComponentDispatch"
};
```

Listing 4 - Mach-O 'thng' Resource

```
// extended 'thng' template
#define thng_RezTemplateVersion 1

#include <Components.r>

resource 'thng' (256) {
        kAQTComponentType,          // Type
        'DEMO',                     // SubType
        'demo',                     // Manufacturer
        0,                          // not used - use componentHasMultiplePlatforms
        0,
        0,
        0,
        'STR ',                     // Name Type
        128,                        // Name ID
        'STR ',                     // Info Type
        129,                        // Info ID
        0,                          // Icon Type
        0,                          // Icon ID
        kMyComponentVersion,        // Version
        componentHasMultiplePlatforms + // Registration Flags
        myComponentRegistrationFlags,
        0,                          // Resource ID of Icon Family
        {
          kMyComponentFlags,
          'dlle',                   // Code Resource type - Entry point found by
                                // symbol name 'dlle' resource
          512,                      // ID of 'dlle' resource
          platformPowerPCNativeEntryPoint,         // Platform Type (response from
                                                // gestaltComponentPlatform or failing
                                                // that, gestaltSysArchitecture)
        };
};
```

Back to top

File Extension and Location

Both kinds of component files should have a ".component" file name extension and be placed in the /Library/QuickTime directory.

Back to top

Carbon and Mac OS 8 / 9 Components

CarbonLib on Mac OS 8 and 9 only supports applications (and in some cases, application plug-ins). It does not support components. Because of this, you will still need to deliver a component that links to InterfaceLib on Mac OS 8 and 9.

You can however, deliver a single binary that contains both the Carbon version of your component linked against CarbonLib for running on Mac OS X and a version linked against InterfaceLib for running on Mac OS 8 and 9.

The platform codes will cause the right version to be loaded on the right platform: platformPowerPCNativeEntryPoint for the Carbon version and platformPowerPC for the non-Carbon version.

Back to top

Sample Code

Electric Image Component Sample - This sample demonstrates the techniques outlined in this document and shows how to build three QuickTime Components; a Graphics Importer, Movie Importer, and Image Decompressor, which all work together to allow QuickTime to use Electric Image format image files. Click here to get the sample.

Back to top

References

For more information regarding 'cfrg' resources consult the CodeFragments.r header file and the Code Fragment Manager chapter of *Inside Macintosh: PowerPC System Software* .

Back to top

Downloadables

Acrobat version of this Note (K). Download

Back to top
