New Technical Notes

Macintosh

Developer Support

Component Manager Q&As

Toolbox

Revised by: Developer Support Center June 1993
Written by: Developer Support Center May 1993

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 for this month:
Typecasting a component for use within OpenComponentResFile
Component Manager is part of System 7.1
Component Manager documentation
Component memory requirements and performance

Typecasting a component for use within OpenComponentResFile

Date Written: 1/20/93
Last reviewed: 4/1/93

How can I typecast a component instance so that it can be used within OpenComponentResFile?

—

When the Component data type is needed from within a component, you can cast the ComponentInstance to type Component and everything will work as expected, as shown below:

```
pascal ComponentResult SomeComponentRoutine(ComponentInstance self)
{
    short  return_val;
```

```
…
if ( return_val = OpenComponentResFile((Component)self) <= 0 )
{
    // Error code here
}
…
```

}

The DrawTextCodec on the QuickTime 1.5 CD-ROM demonstrates the use of various calls such as GetComponentRefcon and OpenComponentResFile from within a component. It's located in the Programming Stuff: Sample Code folder on the CD.

Component Manager is part of System 7.1

Date Written: 1/21/93
Last reviewed: 4/1/93

Is the Component Manager bundled with QuickTime? In other words, will the users of the application I'm designing have to buy QuickTime to get the Component Manager?

—

The Component Manager was first introduced as part of the QuickTime 1.0 system extension. This makes QuickTime a *sine qua non* for system software versions prior to System 7.1. The Component Manager is now part of System 7.1, to give the extended functionality to any Macintosh application running on System 7.1, regardless of the presence or absence of QuickTime.

Component Manager documentation

Date Written: 1/21/93
Last reviewed: 4/1/93

Where can I find documentation on the Component Manager?

—

Presently, the only documentation for the Component Manager is Chapter 4 in the *QuickTime Developer's Guide 1.0.* There will be documentation on the Component Manager in the new *Inside Macintosh* in the future.

There's also an article called "Techniques for Writing and Debugging Components" in issue 12 of *develop.* This article gives you some criteria on whether to write a component and the advantages of using a component. Best of all, there's an accompanying sample code (Dev CD Feb. 93:Periodicals:develop:develop 12 code) for implementing a sample math component.

Component memory requirements and performance

Date Written: 1/21/93
Last reviewed: 4/1/93

I'm concerned with the memory footprint of the Component Manager in a very broad sense. For instance, after closing a tool, does the tool's code resource handle get deallocated? Purged? Marked purgeable? What happens to it? What kind of overhead in both time and space is there for having and calling each "Component Instance"? What kind of system requirements are there? Does the Component Manager work only with System 7?

—

In terms of a component's memory footprint, we trapped on a component and found that it's a purgeable handle. We assume that of course, it would be locked before any actions and unlocked afterwards. So in terms of space overhead, the space would be whatever the size of

the component code resource. But of course, given that the component is purgeable, the Memory Manager would take care of things when necessary.

Going into some specifics about time overhead, the Component Manager has been highly optimized and fast dispatching can reduce its overhead still more, but in general its lookup-and-dispatch process still takes several dozen instructions.

However, there's a fast component dispatch method that eliminates one call to the Component Manager and dispatches directly to your component's entry point. You can then write your component entry point in assembly language. For a detailed instruction on how to do the fast component dispatch method, see page 20 of the article called "Techniques for Writing and Debugging Components" in issue 12 of *develop.*

If the component being called is using the Component Manager's inheritance mechanism, further overhead is incurred passing control to the parent or child component. Overall, the Component Manager is quite efficient, but still not as efficient as direct routine calls. However, note that the Component Manager was originally designed to extend QuickTimes's functionalities in a modular fashion and also satisfy the time constraints that a real-time system such as multimedia would offer. So far, it looks like the time overhead in the QuickTime components hasn't been a problem.

Like everything else, this is an engineering trade-off decision. Components, as supported by the Component Manager, does encourage a modular approach to solve problems. It features inheritance, data hiding, reusability of code, and extensibility. These features needed to be looked at compared to how expensive they are to call and the lack of type checking. Another good article to check out is "Components and C++ classes compared" in *develop* issue 12.

QuickTime Components.p interface conflict
Date Written: 11/30/92
Last reviewed: 3/1/93

When I try to compile the Components.p include file from the QuickTime 1.5 CD, I get "unsatisfied forward reference" error messages. Can you advise?

——

A declaration conflict in the c header files is causing the problem. Until the Components.p file is updated, replace the following declarations:

```
Component = ^privateComponentRecord;
ComponentInstance = ^privateComponentInstanceRecord;

with

Component = ^ComponentRecord
ComponentRecord = RECORD
                  data: ARRAY[0..0] OF LONGINT;
                  END;
```

```
ComponentInstance = ^ComponentInstanceRecord;
ComponentInstanceRecord = RECORD
                          data: ARRAY[0..0] OF LONGINT;
                          END;
```