Technical Note TB505
Component Manager Q&As

CONTENTS

This Technical Note contains a collection of archived Q&As relating to a specific topic--questions sent the Developer Support Center (DSC) along with answers from the DSC engineers. Current Q&A's can be found on the Macintosh Technical Q&A's web site.

[May 01 1993]

SetComponentInstanceA5 used for custom heap

What's the correct usage of `SetComponentInstanceA5`? My component will be registered globally but the individual instances should use the application's storage. The Component Manager chapter says that `SetComponentInstanceA5` is usually called in the open request for a connection. If so, where and how do you get the A5 value of the application to set in the `SetComponentInstanceA5` call?

The Component Manager `SetComponentInstanceA5` (or `GetComponentInstanceA5`) call is used primarily in cases where you might want to set a custom A5 world for your component. In the case that you're using the client's A5 world (default), you don't have to do anything. In fact, A5 will be set to your client's heap for you automatically by the Component Manager. You can call NewHandle in your open routine and it will create a handle in your client's heap. This is your component instances private storage. So, the answer to your question is that you don't have to call `SetComponentInstanceA5` unless you have a custom heap.

However, if you use public globals that you want all component instances to be able to access, you may want to switch to the system zone, allocate the global record with `NewHandle`, and then switch back to the previous zone. The reason you'd want to do this is that you wouldn't want the public globals to disappear if the non-system's heap is disposed of. The reference to public globals is usually stored in the component's refcon.

Back to top

RegisterComponentResource and 'thng' resource

Date Written: 2/17/93

Last reviewed: 5/28/93

I'm using the Component Manager to write my own color picker, and I'm trying to register the component from my own application using `RegisterComponentResource`. The resource fork of the application includes the appropriate resources with the appropriate ID numbers such as `'thng'` and `'cpkr'`, but the register operation fails and I get a null pointer in return. Any suggestions why?

The documentation is somewhat obscure on this, but what you need to pass to `RegisterComponentResource` is the `'thng'` resource. Depending on what you're doing, you may want to pass a value of 1 in the global parameter, to make your component available to other applications. The following code accomplishes what you want:

```
Component RegisterPicker (short pickerID)
{
    Component    aComponent = 0L;    /* prepare for failure */
    ComponentResource    **CmpHdl;
    CmpHdl = (ComponentResource **) GetResource ('thng', pickerID);
    if (CmpHdl)                 /* make sure you got the resource */
        aComponent = RegisterComponentResource (CmpHdl, 0);
    return ( aComponent );
```

Back to top

Component Manager and multiple segments

Date Written: 2/10/93

Last reviewed: 7/2/93

How does the Component Manager handle multiple segments?

The Component Manager doesn't automatically support multiple code segments. In fact, it assumes that all the code is in one segment. The `'thng'` resource allows you to specify one segment. Therefore, if you load other code segments from your main segment, they should be loaded and locked in the system heap. There are several ways you can do this; the register routine probably is the best place to do it. Since the register routine is called only once at registration time, this allows your component to completely load the remaining code segments into the system heap. Components should ensure that A5 (or A4) is set appropriately before any intersegment calls. For more information, see "Managing Component Registration" in issue 15 of *develop.*

Back to top

Typecasting a component for use within OpenComponentResFile

Date Written: 1/20/93

Last reviewed: 4/26/93

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 component instance 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
    }
    . . .
```

`DrawTextCodec` on the QuickTime 1.5 CD demonstrates the use of various calls such as `GetComponentRefcon` and `OpenComponentResFile` from within a component. It's also located on the Developer CD Series disc, Tool Chest edition.

Back to top

Component Manager is part of System 7.1

Date Written: 1/21/93

Last reviewed: 7/2/93

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

The Component Manager was first introduced as part of the QuickTime 1.0 system extension. This makes the presence of QuickTime a requirement for system software versions earlier than System 7.1. The Component Manager is now part of System 7.1, giving the extended functionality to any Macintosh application running on System 7.1 regardless of the presence of QuickTime.

Back to top

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.

Back to top

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;
```

Back to top

Downloadables

Acrobat version of this Note (K). Download
