Technical Note TB555
Resource Manager Q&As

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.

[Sep 01 1993]

Order of resources retrieved by GetIndResource

When `GetIndResource` is called, does the Resource Manager return the resource with the lowest ID? Is it possible to predict the order of resources returned by `GetIndResource` call?

`GetIndResource` returns resources in the order that they were added to the resource file, not by ID number. If you want to retrieve resources in a specific order by ID number, use `GetIndResource` and `GetResInfo` *(Inside Macintosh* Volume I, page 121) to build a list of resources and ID numbers, and then sort the list and start retrieving based on your criteria.

Theoretically, if you add resources in a certain order, `GetIndResource` will give them back to you in the same order. However, this is subject to change, which is why we recommend the above approach.

Back to top

Macintosh resource ID numbering

Date Written: 7/26/90

Last reviewed: 6/14/93

Does the warning about using Macintosh resource IDs 0 through 127 apply to resource types of my own design?

When numbering resources, you can use any number you like on resource types of your own creation (that is, not of a type reserved by Apple.)

In numbering resources you create of a type reserved by Apple, stick to the following guidelines:

- Negative numbers are reserved for various owned and otherwise "encoded" resources.
- The numbers 0-127 are reserved for Apple's use.
- Positive numbers >127 are available for your use.
- DTS no longer registers `'FOND'` IDs. `'FOND'`s should be referenced by name and not by number because the numbers change. Choose any `'FOND'` ID within your font's script range. For details, see the Macintosh Technical Note "Font Family Numbers."

Back to top

Locate Macintosh folder first with FindFolder before OpenRFPerm

Date Written: 12/4/90

Last reviewed: 12/19/90

`OpenRFPerm` gets result code -43 (file not found) when opening a file that was put in the Macintosh System 7 Control Panel folder, but when the file is moved outside the Control Panel folder and within the System Folder, `OpenRFPerm` works fine.

It sounds like you are calling `OpenRFPerm` and it is accessing the System Folder rather than the Control Panel Folder. System 7.0 doesn't automatically search all of the subfolders in the System Folder; therefore, it is necessary to use the toolbox call `FindFolder` to locate the Control Panel folder before you call `OpenRFPerm`.

The `FindFolder` call is documented in *Inside Macintosh* Volume VI, Chapter 8: "You can call the FindFolder function to get the path information so that you can access special folders. You pass FindFolder a target volume and a constant that tells it which special folder you are interested in. FindFolder returns a volume reference number and a directory ID. If the specified folder does not exist, FindFolder can create it and return the new directory ID. (See table 8-2 for the folder types [in System 7.0], resource types and constants.)"

Back to top

HOpenResFile with fsRdPerm permission returns unique path

Date Written: 1/21/91

Last reviewed: 2/13/91

According to the Macintosh Technical Note "OpenRFPerm: What your mother never told you," "OpenRFPerm will create multiple, unique, read-only access paths to a resource file." Is this same behavior present in `HOpenResFile`? A cdev I have written must look within certain types of files for specific resources. If the file is already open, will a call to `HOpenResFile` with `fsRdPerm` permission return a unique path? Using `OpenRFPerm` seems to have no problems, but the overhead of creating working directories seems rather untidy.

`HOpenResFile` is simply a friendly face on top of `OpenRFPerm`, so the unique access paths that are created by `OpenRFPerm` will indeed percolate upward through `HOpenResFile`.

Back to top

Changing a Macintosh resource

Date Written: 4/15/91

Last reviewed: 6/14/93

What's the recommended way to update a Macintosh resource?

As you may have guessed, there is no standard way to update a resource in a resource file. The general rule of thumb is, if the resource exists, simply load it in, modify it, mark it changed, and then cause it to be written out. It is also best at this point to write the resource if you know you are done changing it (`ChangedResource` simply schedules the resource to be written the next time the resource file is updated). (Read the caveat about changing purgeable resource on page I-123 of *Inside Macintosh* Volume I.)

You should, as a general rule, never remove and replace the resource. This causes undue accounting problems for yourself and the Resource Manager, and causes quite a bit more writing to the file than is absolutely needed.

If the write of a resource has failed for any reason and you would like to have the memory version of the resource contain what the disk version does, simply detach the current handle from the resource using detach, dispose of the handle if you don't want the data anymore, and reload the resource as normal.

Back to top

System 7.0 Resource Manager resource decompression

Date Written: 4/26/91

Last reviewed: 6/14/93

Is the resource compression technique used in System 7.0 documented anywhere?

Documentation describing the compression algorithm used on resources is not available. The System 7.0 Resource Manager automatically decompresses for you. You can force a resource to be permanently decompressed by modifying it with ResEdit. Under System 6.0.7 or earlier systems, there's no easy method to access compressed resources.

Back to top

HOpenResFile versus OpenResFile

Date Written: 5/1/91

Last reviewed: 6/7/91

Why would using `OpenResFile`(fileName) cause a crash when I try to open a Macintosh font file that's already open?

The problem stems from the fact that `OpenResFile` doesn't deal effectively with cases where the resource file is already open. Luckily, there are some relatively new Resource Manager calls that you can and should use instead. They're all documented in the Resource Manager chapter of *Inside Macintosh* Volume VI and in the Macintosh Technical Note "New Resource Manager Calls."

The call of interest in your case is `HOpenResFile`. To use it, break down the `vRefNum` (actually `WDRefNum`) returned by Standard File into a real `vRefNum` and `dirID` by calling `PBGetWDInfo`, and pass those to `HOpenResFile` along with the file name. The important part, however, is the permissions byte. If you expect to modify the file, pass `fsRdWrPerm` in that field. If there's an error of any kind, expect `HOpenResFile` to return -1, which should serve as a signal that you need to call `ResError` to find out what went wrong.

Back to top

Partial resources and compressed resource format

Date Written: 8/9/91

Last reviewed: 9/24/91

The partial resource calls `ReadPartialResource` and `WritePartialResource` do not decompress Apple's compressed resource format. In fact, these two calls are the only way possible to read resources without decompressing them. It is not really possible to decompress part of a resource, since the decompression method requires the entire resource to be in memory for the conversion to occur.

Details of Apple's compression scheme are not documented and Apple doesn't currently license the algorithm to developers, but Macintosh compressed resources are contained only in the System file and other Apple-specific files. Because third-party products don't use Apple's compressed resources and all Apple's compressed resources fit in memory without the partial calls, you needn't worry about supporting partial resource calls with compressed resources. Just use the standard `GetResource` call to load these Apple-compressed resources. You can determine if a resource is compressed by examining bit 0 of the resource's attributes. If this bit is set, the resource is compressed, and partial resource calls will return the compressed data.

Back to top

Getting Macintosh system strings

Date Written: 1/18/92

Last reviewed: 2/6/92

Our application needs to know the Chooser Name and Machine Name, stored as `'STR '` resources in the System file. If I do a `GetResource` to read one of these (they are loaded into the system heap), should I do a `ReleaseResource` when I am done? The resources are not purgeable so they won't go away by themselves. On the other hand, I'm worried about releasing them out from under something else that has the same resource handle (like the Chooser or the Finder).

The Chooser Name and Machine Name resources should be treated a little differently from normal resources under System 7. You really should use `GetString`, instead of `GetResource` to get the string (although `GetString` simply calls `GetResource`). Once you have the string you should not release, dispose or make purgeable the string. You will find that the string was already loaded when you asked for it, so it should remain when you are done. Also, you should never change the contents of, or mark as changed, either of these strings since the Chooser could be open and would not recognize your changes, unless you give your users sufficient warning of the potentially confusing side effects of having a different name in two places. Keep in mind that System 6 does not necessarily have the Computer Name resource, so do error checking...

Bottom line: Get but don't release these special strings, and don't modify them unless absolutely needed.

Back to top

Pre-load resources & calling OpenResFile on another application

Date Written: 2/27/92

Last reviewed: 6/12/92

My application wants to open other applications and play with the resources therein, like ResEdit, but when it calls OpenResFile on an application, the program gets lost in `GetNamedResource`. Is there something I'm missing?

Your problem stems from the fact that some resources in the application file you're opening with `OpenResFile` are marked to be preloaded, and so are loaded into memory when the resource fork is opened.

Since most applications have CODE resources marked to be preloaded, this turns into a much bigger problem, because the Segment Loader will treat these preloaded CODE resources as your code resources if you make a between-segment call that triggers a call to `LoadSeg` while the opened resource file is first in the resource chain. If this happens, you'll begin executing code out of the other application, which will cause your Macintosh to crash and burn.

The solution to this problem is to bracket `OpenResFile` calls with `SetResLoad(FALSE)` and `SetResLoad(TRUE)`, and to avoid making between-segment calls when you've got another resource file open that contains CODE resources. This will not only prevent your application's memory from being used by preloaded resources that you don't want, but will also prevent the Segment Loader from jumping into the other application's code. If you need to get CODE resources out of the opened resource file, you can still prevent the Segment Loader problem by calling `UseResFile` on your application's resource reference number to put your application at the top of the resource chain.

Back to top

Maximum number of items in a Macintosh resource file is 2727

Date Written: 5/22/90

Last reviewed: 12/17/90

Is there a maximum number of items for Macintosh resources?

In a file, yes. It is 2727.

X-Ref:

Macintosh Technical Note "Maximum Number of Resources in a File"

Back to top

Maximum Macintosh resource size is "maxlongint" bytes

Date Written: 5/22/90

Last reviewed: 12/17/90

Is the maximum size of a Macintosh resource still 32K?

No. There used to be a bug in the 64K ROMs that didn't allow you to write even multiples of 32K, such as 32K-64K or 128K-192K. This bug was fixed in 128K ROMs. As of 128K ROMs, the resource size is limited to `maxlongint` bytes.

Back to top

Downloadables

	Acrobat version of this Note (K).	Download

Technical Notes by Date | Number | Technology | Title
Developer Documentation | Technical Q&As | Development Kits | Sample Code