New Technical Notes
Macintosh

Developer Support

Edition Manager Q&As
Interapplication Communication

Revised by: Developer Support Center October 1992
Written by: Developer Support Center October 1990

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.

Ordering of 'fmts' in an edition container
Date Written: 6/13/91
Last reviewed: 8/1/92

Is there any ordering to the 'fmts' edition format, like best to worst? For applications that support subscribing to a large number of formats, defaulting to "best" is nontrivial.

—

There is no implicit ordering of the formats in an edition container. They are written and read and returned in the 'fmts' resource in the order in which they were written into the container. This is not an arbitrary model, it's based on information that the Edition Manager, as system software, _can't_ know. The decision would be _very_ implementation dependent. Your users may prefer your own custom edition type over all others, so the way you would want the 'fmts' resource to be returned would be a great deal different than a simple word processor, who would be preferring TEXT.

So the system can't make that decision. It doesn't know who is calling the edition manager and can't know what format your specific application prefers. Any ordering it may make would be wrong for a large set of developers. So, it relies on just giving back the formats in the order that they were written. Of course, if you are customizing your Subscriber dialog

with your own popup, Apple recommends that you order the popup (or list, or whatever) to suit the needs of your application best, defaulting to your “best” format; however, there is no way to enforce this. The enhanced subscriber dialogs design is/are up to individual developers.

SetEditionFormatMark's setMarkTo parameter
Date Written: 7/29/91
Last reviewed: 8/1/92

How is SetEditionFormatMark's third parameter (setMarkTo) to be determined? Is it always zero for the start of each format? Does it mean that any chunk of the edition can be obtained, using the starting mark and byte count, exactly like reading a file?

___

The setMarkTo parameter is set only for the specific format you're concerned with. So yes, it is always initially zero for any specific format in an edition container file. And you can set it to whatever you like for that particular format. You can get the length of the format you're concerned with by calling

```
EditionHasFormat(refNum,formatDesired,&returnedSize);
```

and returnedSize will contain the size of that format. So in SetEditionFOrmatMark your range is 0-returnedSize.

The answer to your third question is "kinda." You can almost think of the separate editions in a edition container file as accessible the same way a normal data fork is, but you have to remember that you must work with the Edition Manager calls to make sure this works.

The edition container format mark for a format does NOT relate at all to the actual mark the File Manager is using in the data fork. The Edition Manager does its own translation from Edition Manager mark to logical file mark when you make a ReadEdition call. So, you are never informed of where the actual data resides in the ECF.

This is necessary to allow the Edition Manager room to expand. By having the Edition Manager make the conversions, you achieve two things:

1) The header in the ECF can be modified, extended or shortened anytime, and no applications are dependent on the "real" file mark.

2) You as an application don't have to worry about another format being added to the ECF while you are reading another one. Say the EDF currently has two formats, a 1K PICT at the beginning of the file, and a 2K TEXT after it. You are reading the TEXT 50 bytes at a time (for some reason or other). While you are doing that, the publisher updates the 'PICT'. The TEXT is held in a buffer, because the EM knows that someone is reading it. But the PICT can be written now, even if its size changes, and the Edition Manager just changes its offset to the mark for the TEXT. You as the application still continue to use the same mark; the size of the file changed but you did not have to be aware of it.

So you can manage reading formats yourself, you can "batch" in a large format (or write it

out in chunks) but you have to use Edition Manager read and write calls to do it, not File Manager calls.

Apple 'cncl' section cancel event

Date Written: 8/23/91
Last reviewed: 8/1/92

When will the “section cancel” Apple event be implemented in the future so the system can send it? I’ve implemented it, but because it’s hard to tell it, I’m tempted to take it out, unless you can give me a good reason to leave it in.

—

There’s no indication when this event will be implemented by the system. The 'cncl' event is primarily in there for applications to facilitate scripting compatibility, not for system use. Since Apple is encouraging all applications to send Apple events to themselves in response to user actions, the 'cncl' event is provided for completeness and user understanding. When scripting systems become more prevalent, then the 'cncl' (and other events) will make a lot more sense, since a scripter can then record (or a user scripting tool generate) a section cancel event and keep it in an Apple event script.

Checking for registered sections
Date Written: 10/23/91
Last reviewed: 8/1/92

Inside Macintosh states. “…your application must keep its own list of registered sections for each open document that contains sections.” Which NewSection() and RegisterSection() results codes indicate the section was not registered successfully? Is there a field in the section record which can be examined to determine if the section is registered?

—

Checking the error codes is always a good idea, but the IsRegisteredSection routine should be used to determine if a section is registered—for example, after calling NewSection and RegisterSection. The fields of the section record should not be used to determine this.