

New Technical Notes

Macintosh



®

Developer Support

Image Compression Manager Q&As QuickTime

Revised by: Developer Support Center

June 1993

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.

New Q&As in this Technical Note:

Using QuickTime dither tables in a codec

How to tell whether a picture is QuickTime-compressed

Using QuickTime dither tables in a codec

Date Written: 1/25/93

Last reviewed: 4/1/93

How can I use QuickTime fast dither tables provided by the Image Compression Manager to write a codec? I haven’t been able to find any documentation on how to access and use them. Are these tables available?

For QuickTime 1.0 you could use the `MakeDitherTable` and `DisposeDitherTable` calls in `ImageCompression.h`. The calls were taken out for QuickTime 1.5 because the format is likely to change and you’d break in the future. The current dither table format isn’t available for that reason, though the documentation on the QuickTime 1.0 CD describes the calls if that helps.

You can use QuickTime to perform the dithering. If you do use QuickTime, you could draw

the image in an off-screen GWorld, using the DrawPictureFile with the dither flag set, and then compress it with your codec.

How to tell whether a picture is QuickTime-compressed

Date Written: 12/2/92

Last reviewed: 4/1/93

How can I tell whether or not a picture is QuickTime-compressed?

The key to your question is “sit in the bottlenecks.” If the picture contains any QuickTime-compressed images, the images will need to pass through the StdPix bottleneck. This is a new graphics routine introduced with QuickTime. Unlike standard QuickDraw images, which only call StdBits, QuickTime-compressed images need to be decompressed first in the StdPix routine. Then QuickDraw uses StdBits to render the decompressed image. So, swap out the QuickDraw bottlenecks, and put some code in the StdPix routine. If it’s called when you call DrawPicture, you know you have a compressed picture. To determine the type of compression, you can access the image description using GetCompressedPixMapInfo. The cType field of the ImageDescription record will give you the codec type. See the Snippets: Imaging: Graphics: CollectPictColors snippet and page 46-47 of develop Issue 13 for further reference on swapping out the bottlenecks.

Saving offscreen GWorld as compressed PICT resource

Date Written: 11/6/92

Last reviewed: 3/1/93

Can I use the CompressPicture routine to spool in a source picture from disk by overriding the QuickDraw proc getPicProc as documented in *Inside Macintosh* Volume V, pages 88-89? I’m trying to save the contents of an off-screen GWorld as a compressed PICT resource. Unfortunately there’s no direct way to compress the GWorld’s pixMap to a resource.

We definitely don’t recommend trying to spool in or out the results of CompressPicture or CompressImage. We recommend doing one of the following instead:

- You can compress the GWorld using CompressImage and then call OpenPicture, DecompressImage, and ClosePicture using a data-unloading picture proc. The drawback here is that you need to have a copy of the compressed image in memory.
- If it’s unacceptable to have an entire compressed image in memory, you can consider banding along with data unloading. So, you’d then call OpenPicture, CompressImage on a band, DecompressImage on a band, CompressImage on another band, DecompressImage on the other band, and so on. When all bands are done, then call ClosePicture. The drawback for this is that the compressed picture will have bands of image data that won’t display well dithered. This could be an issue, but the best way to find out is to try it.

The second suggestion is probably the best idea if you want to keep your memory footprint small. But much of the decision depends on your application.

QuickTime fills in image descriptor when data is compressed

Date Written: 6/14/91

Last reviewed: 10/22/91

When I send compressed images over Ethernet, CompressSequenceBegin doesn't fill in the ImageDescription, which is needed at the other end of the conference link to DecompressSequenceBegin. Is this a bug?

CompressSequenceBegin doesn't actually modify the handle that you pass. Instead, QuickTime makes a note of the handle that's passed and doesn't actually modify the contents until the first call that actually compresses data, such as CompressSequenceFrame. At that point, the handle will be changed.

If you can postpone dealing with the image descriptor until after the first call that compresses data, whatever you are writing should work just fine.

Decompressing to partial window: Bug & workaround

Date Written: 6/18/92

Last reviewed: 9/15/92

Under System 7, decompressing directly to a window that is partially "off the screen" (that is, not completely visible) results in a -50 (invalid param) QuickTime error. We can special case when windows are off the screen and decompress into an offscreen GWorld but we would prefer a fix to either QuickTime or System 7.

The problem you are having is due to a bug in the Image Compression Manager. It fails to clear QDError when starting a decompression job and later checks it to see if it is OK to continue the operation. Something else is setting QDerr and your call fails.

The solution that you can implement now consists of clearing QDerr before calling any of the decompression routines. You can accomplish this by calling QDError (which clears the error after it passes the current value to you) or zeroing the low mem QDerr (0xD6E) by hand.

Future versions of QuickTime will have the fix and will not require that you work around the problem.