
QCPlugInContext Protocol Reference

[Cocoa > Graphics & Imaging](#)



2007-05-09



Apple Inc.
© 2007 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, Cocoa, Mac, Mac OS, and Quartz are trademarks of Apple Inc., registered in the United States and other countries.

OpenGL is a registered trademark of Silicon Graphics, Inc.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

QCPluginContext Protocol Reference 5

Overview 5

Tasks 5

 Getting the OpenGL Context 5

 Logging Messages 5

 Getting Execution Context Information 5

 Getting an Image Provider 6

Instance Methods 6

 bounds 6

 CGLContextObj 6

 colorSpace 7

 logMessage: 7

 outputImageProviderFromBufferWithPixelFormat:pixelsWide:pixelsHigh:baseAddress:
 bytesPerRow:releaseCallback:releaseContext:colorSpace:shouldColorMatch: 8

 outputImageProviderFromTextureWithPixelFormat:pixelsWide:pixelsHigh:name:
 flipped:releaseCallback:releaseContext:colorSpace:shouldColorMatch: 9

 userInfo 10

Document Revision History 11

Index 13

QCPlugInContext Protocol Reference

Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	QCPlugIn.h

Overview

The `QCPlugInContext` protocol defines methods that you use only from within the execution method (`execute:atTime:withArguments:`) of a `QCPlugIn` object.

Tasks

Getting the OpenGL Context

- [CGLContextObj](#) (page 6)
Returns the destination CGL context to use for OpenGL rendering from within the execution method.

Logging Messages

- [logMessage:](#) (page 7)
Writes a message to the Quartz Composer log.

Getting Execution Context Information

- [userInfo](#) (page 10)
Returns a mutable dictionary that contains information that can be shared between all instances of the `QCPlugIn` subclass, running in the same Quartz Composer context.
- [bounds](#) (page 6)
Returns the bounds of the rendering context.
- [colorSpace](#) (page 7)
Returns the color space used by the rendering context.

Getting an Image Provider

- [outputImageProviderFromBufferWithPixelFormat:pixelSwide:pixelShigh:baseAddress:bytesPerRow:releaseCallback:releaseContext:colorSpace:shouldColorMatch:\(page 8\)](#)

Returns an image provider from a single memory buffer.

- [outputImageProviderFromTextureWithPixelFormat:pixelSwide:pixelShigh:name:flipped:releaseCallback:releaseContext:colorSpace:shouldColorMatch:\(page 9\)](#)

Returns an image provider from an OpenGL texture.

Instance Methods

bounds

Returns the bounds of the rendering context.

- (NSRect) bounds

Return Value

The bounds of the rendering context expressed in Quartz Composer units.

Availability

Available in Mac OS X v10.5 and later.

Declared In

QCPlugIn.h

CGLContextObj

Returns the destination CGL context to use for OpenGL rendering from within the execution method.

- (CGLContextObj) CGLContextObj

Return Value

The destination CGL context.

Discussion

To send commands to the OpenGL context:

- Use CGL macros instead of changing the current OpenGL context.
- Save and restore all OpenGL states except those defines by `GL_CURRENT_BIT` (vertex position, color, texture, and so on)

The following code shows how you'd use the method `CGLContextObj`:

```
// Set up using CGL macros.
#import <OpenGL/CGLMacro.h>

- (BOOL) execute:(id<QCPlugInContext>)context
             atTime:(NSTimeInterval)time
       withArguments:(NSDictionary *)arguments
```

```

{
    // Set the CGL context to a local variable.
    CGLContextObj cgl_ctx = [context CGLContextObj];
    if(cgl_ctx == NULL)
        return NO;

    // Save and set OpenGL states.
    // Put your OpenGL code here.
    // Restore the OpenGL states.
    return YES;
}

```

You can retrieve the corresponding OpenGL pixel format by calling the function `CGLGetPixelFormat`.

Availability

Available in Mac OS X v10.5 and later.

Declared In

QCPlugIn.h

colorSpace

Returns the color space used by the rendering context.

- (CGColorSpaceRef) colorSpace

Return Value

An RGB color space; NULL if the custom patch execution mode is not consumer.

Discussion

If the method returns a color space, it must be an RGB color space.

Availability

Available in Mac OS X v10.5 and later.

Declared In

QCPlugIn.h

logMessage:

Writes a message to the Quartz Composer log.

- (void) logMessage:(NSString*)format, ...

Parameters

format

The string to write to the log. The default location for the log is the standard output.

Discussion

This method is an alternative to using the functions `NSLog` or `printf`.

Availability

Available in Mac OS X v10.5 and later.

Declared In
QCPlugIn.h

outputImageProviderFromBufferWithPixelFormat:pixelsWide:pixelsHigh:baseAddress:bytesPerRow:releaseCallback:releaseContext:colorSpace:shouldColorMatch:

Returns an image provider from a single memory buffer.

```
- (id) outputImageProviderFromBufferWithPixelFormat:(NSString*)format
    pixelsWide:(NSUInteger)width pixelsHigh:(NSUInteger)height baseAddress:(const
    void*)baseAddress bytesPerRow:(NSUInteger)rowBytes
    releaseCallback:(QCPlugInBufferReleaseCallback)callback
    releaseContext:(void*)context colorSpace:(CGColorSpaceRef)colorSpace
    shouldColorMatch:(BOOL)colorMatch
```

Parameters

format

The pixel format of the memory buffer. This must be compatible with the color space.

width

The width, in bytes, of the memory buffer.

height

The height, in bytes, of the memory buffer.

baseAddress

The base address of the memory buffer, which must be multiple of 16.

rowBytes

The number of bytes per row of the memory buffer, which must be multiple of 16.

callback

The release callback. Your callback must use this type definition:

```
typedef void (*QCPlugInBufferReleaseCallback)(const void* address, void* context);
```

If you name your callback function `MyQCPlugInBufferReleaseCallback`, you would declare it like this:

```
void MyQCPlugInBufferReleaseCallback (const void address,
    void * context);
```

Quartz Composer invokes your callback when the memory buffer is no longer needed. The callback can be called from any thread at any time

context

The context to pass to the release callback.

colorSpace

The color space of the memory buffer. This must be compatible with the pixel format.

colorMatch

A Boolean that specifies whether Quartz Composer should color match the image. Pass NO if the image is a mask or gradient or should not be color matched for some other reason. Otherwise, pass YES.

Return Value

An image provider.

Discussion

You must not modify the image until the release callback is invoked.

Availability

Available in Mac OS X v10.5 and later.

Declared In

QCPlugIn.h

outputImageProviderFromTextureWithPixelFormat:pixelsWide:pixelsHigh:name:flipped:releaseCallback:releaseContext:colorSpace:shouldColorMatch:

Returns an image provider from an OpenGL texture.

```
- (id) outputImageProviderFromTextureWithPixelFormat:(NSString*)format
    pixelsWide:(NSUInteger)width pixelsHigh:(NSUInteger)height name:(GLuint)name
    flipped:(BOOL)flipped releaseCallback:(QCPlugInTextureReleaseCallback)callback
    releaseContext:(void*)context colorSpace:(CGColorSpaceRef)colorSpace
    shouldColorMatch:(BOOL)colorMatch;
```

Parameters

format

The pixel format of the texture. This must be compatible with the color space.

width

The width, in bytes, of the texture.

height

The height, in bytes, of the texture.

name

An OpenGL texture of type `GL_TEXTURE_RECTANGLE_EXT` that is valid on the Quartz Composer OpenGL context. Note that textures do not have a retain and release mechanism. This means that your application must make sure that the texture exists for the life cycle of the image provider.

flipped

YES to have Quartz Composer flip the contents of the texture vertically.

callback

The release callback. Your callback must use this type definition:

```
typedef void (*QCPlugInTextureReleaseCallback)(CGLContextObj cgl_ctx, GLuint
name, void* context);
```

If you name your callback function `MyQCPlugInTextureReleaseCallback`, you would declare it like this:

```
void MyQCPlugInTextureReleaseCallback (CGLContextObj cgl_ctx,
    GLuint name,
    void* context);
```

Quartz Composer invokes your callback when the memory buffer is no longer needed. The callback can be called from any thread at any time

context

The context to pass to the release callback.

colorSpace

The color space of the texture. This must be compatible with the pixel format.

colorMatch

A Boolean that specifies whether Quartz Composer should color match the texture. Pass `NO` if the texture is a mask or gradient or should not be color matched for some other reason. Otherwise, pass `YES`.

Return Value

An image provider.

Discussion

You must not modify the texture until the release callback is invoked.

Availability

Available in Mac OS X v10.5 and later.

Declared In

QCPlugIn.h

userInfo

Returns a mutable dictionary that contains information that can be shared between all instances of the `QCPlugIn` subclass, running in the same Quartz Composer context.

- (`NSMutableDictionary*`) `userInfo`

Return Value

A mutable dictionary.

Discussion

When you add information to the dictionary, make sure that you use unique keys, such as `com.myCompany.foo`. You can use this dictionary to cache data that you want to share.

Availability

Available in Mac OS X v10.5 and later.

Declared In

QCPlugIn.h

Document Revision History

This table describes the changes to *QCPlugInContext Protocol Reference*.

Date	Notes
2007-05-09	New document that describes the protocol used by execution contexts.

REVISION HISTORY

Document Revision History

Index

B

bounds [protocol instance method 6](#)

C

CGLContextObj [protocol instance method 6](#)

colorSpace [protocol instance method 7](#)

L

logMessage: [protocol instance method 7](#)

O

outputImageProviderFromBufferWithPixelFormat:
pixelsWide:pixelsHigh:baseAddress:bytesPerRow:
releaseCallback:releaseContext:colorSpace:
shouldColorMatch: [protocol instance method 8](#)

outputImageProviderFromTextureWithPixelFormat:
pixelsWide:pixelsHigh:name:flipped:
releaseCallback:releaseContext:colorSpace:
shouldColorMatch: [protocol instance method 9](#)

U

userInfo [protocol instance method 10](#)