Quartz Display Services Reference

Graphics & Imaging > Quartz

2008-11-19

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

Simultaneously published in the United States and Canada.

Contents

Quartz Display Services Reference

Framework:	ApplicationServices/ApplicationServices.h
Companion guide	Quartz Display Services Programming Topics
Declared in	CGDirectDisplay.h CGDirectPalette.h CGDisplayConfiguration.h CGDisplayFade.h CGError.h CGRemoteOperation.h CGSession.h CGWindowLevel.h

Overview

Note: This document was previously titled *Quartz Services Reference*. Some information related to low-level events has been moved from this document into *Quartz Event Services Reference*.

Quartz Display Services provides direct access to certain low-level features in the Mac OS X window server related to the configuration and control of display hardware. For example, you can use Quartz Display Services to:

- Examine and change display mode properties such as width, height, and pixel depth
- Configure a set of displays in a single operation
- Capture one or more displays for exclusive use
- Perform fade effects
- Activate display mirroring
- Configure gamma color correction tables and color palettes
- Receive notification of screen update operations

Functions by Task

Finding Displays

`CGMainDisplayID` (page 60)
Returns the display ID of the main display.

`CGGetOnlineDisplayList` (page 59)
Provides a list of displays that are online (active, mirrored, or sleeping).

`CGGetActiveDisplayList` (page 53)
Provides a list of displays that are active (or drawable).

`CGGetDisplaysWithOpenGLDisplayMask` (page 54)
Provides a list of displays that corresponds to the bits set in an OpenGL display mask.

`CGGetDisplaysWithPoint` (page 55)
Provides a list of online displays with bounds that include the specified point.

`CGGetDisplaysWithRect` (page 56)
Gets a list of online displays with bounds that intersect the specified rectangle.

`CGOpenGLDisplayMaskToDisplayID` (page 60)
Maps an OpenGL display mask to a display ID.

`CGDisplayIDToOpenGLDisplayMask` (page 36)
Maps a display ID to an OpenGL display mask.

Capturing and Releasing Displays

`CGDisplayCapture` (page 31)
Captures a display for exclusive use by an application.

`CGDisplayCaptureWithOptions` (page 31)
Captures a display for exclusive use by an application, using the specified options.

`CGDisplayRelease` (page 45)
Releases a captured display.

`CGDisplayIsCaptured` (page 39)
Returns a Boolean value indicating whether a display is captured.

`CGCaptureAllDisplays` (page 16)
Captures all attached displays.

`CGCaptureAllDisplaysWithOptions` (page 17)
Captures all attached displays, using the specified options.

`CGReleaseAllDisplays` (page 67)
Releases all captured displays.

`CGShieldingWindowID` (page 73)
Returns the window ID of the shield window for a captured display.

`CGShieldingWindowLevel` (page 74)
Returns the window level of the shield window for a captured display.

`CGDisplayAddressForPosition` (page 23)
Returns the address in frame buffer memory that corresponds to a position on an online display.

`CGDisplayBaseAddress` (page 25)
Returns the base address in frame buffer memory of an online display.

`CGDisplayGetDrawingContext` (page 35)
Returns a graphics context suitable for drawing to a captured display.

Configuring Displays

`CGBeginDisplayConfiguration` (page 15)
Begins a new set of display configuration changes.

`CGCancelDisplayConfiguration` (page 16)
Cancels a set of display configuration changes.

`CGCompleteDisplayConfiguration` (page 17)
Completes a set of display configuration changes.

`CGConfigureDisplayMirrorOfDisplay` (page 19)
Changes the configuration of a mirroring set.

`CGConfigureDisplayMode` (page 20)
Configures the display mode of a display.

`CGConfigureDisplayOrigin` (page 21)
Configures the origin of a display in global display (desktop) coordinates.

`CGRestorePermanentDisplayConfiguration` (page 69)
Restores the permanent display configuration settings for the current user.

`CGConfigureDisplayStereoOperation` (page 22)
Enables or disables stereo operation for a display, as part of a display configuration.

`CGDisplaySetStereoOperation` (page 49)
Immediately enables or disables stereo operation for a display.

Getting the Display Configuration

`CGDisplayCopyColorSpace` (page 32)
Returns the color space for a display.

`CGDisplayIOServicePort` (page 36)
Returns the I/O Kit service port of the specified display.

`CGDisplayIsActive` (page 37)
Returns a Boolean value indicating whether a display is active.

`CGDisplayIsAlwaysInMirrorSet` (page 37)
Returns a Boolean value indicating whether a display is always in a mirroring set.

`CGDisplayIsAsleep` (page 38)
Returns a Boolean value indicating whether a display is sleeping (and is therefore not drawable.)

`CGDisplayIsBuiltin` (page 38)
Returns a Boolean value indicating whether a display is built-in, such as the internal display in portable systems.

`CGDisplayIsInHWMirrorSet` (page 39)
Returns a Boolean value indicating whether a display is in a hardware mirroring set.

`CGDisplayIsInMirrorSet` (page 40)
Returns a Boolean value indicating whether a display is in a mirroring set.

`CGDisplayIsMain` (page 40)
Returns a Boolean value indicating whether a display is the main display.

`CGDisplayIsOnline` (page 41)
Returns a Boolean value indicating whether a display is connected or online.

`CGDisplayIsStereo` (page 41)
Returns a Boolean value indicating whether a display is running in a stereo graphics mode.

`CGDisplayMirrorsDisplay` (page 42)
For a secondary display in a mirroring set, returns the primary display.

`CGDisplayModelNumber` (page 42)
Returns the model number of a display monitor.

`CGDisplayPrimaryDisplay` (page 44)
Returns the primary display in a hardware mirroring set.

`CGDisplayRotation` (page 46)
Returns the rotation angle of a display in degrees.

`CGDisplayScreenSize` (page 47)
Returns the width and height of a display in millimeters.

`CGDisplaySerialNumber` (page 48)
Returns the serial number of a display monitor.

`CGDisplayUnitNumber` (page 51)
Returns the logical unit number of a display.

`CGDisplayUsesOpenGLAcceleration` (page 52)
Returns a Boolean value indicating whether Quartz is using OpenGL-based window acceleration (Quartz Extreme) to render in a display.

`CGDisplayVendorNumber` (page 52)
Returns the vendor number of the specified display's monitor.

Registering for Notification of Display Configuration Changes

These functions are used to register and unregister a callback function for notification of display configuration changes.

`CGDisplayRegisterReconfigurationCallback` (page 45)
Registers a callback function to be invoked whenever a local display is reconfigured.

`CGDisplayRemoveReconfigurationCallback` (page 46)
Removes the registration of a callback function that’s invoked whenever a local display is reconfigured.

Retrieving Display Parameters

`CGDisplayBounds` (page 29)
Returns the bounds of a display in global display space.

`CGDisplayPixelsHigh` (page 43)
Returns the display height in pixel units.

`CGDisplayPixelsWide` (page 44)
Returns the display width in pixel units.

`CGDisplayBitsPerPixel` (page 29)
Returns the number of bits used to represent a pixel in the frame buffer.

`CGDisplayBitsPerSample` (page 29)
Returns the number of bits used to represent a pixel component in the frame buffer.

`CGDisplaySamplesPerPixel` (page 47)
Returns the number of color components used to represent a pixel.

`CGDisplayBytesPerRow` (page 30)
Returns the number of bytes per row in a display.

Using Display Modes

`CGDisplayAvailableModes` (page 24)
Returns information about the currently available display modes.

`CGDisplayBestModeForParameters` (page 26)
Returns information about the display mode closest to a specified depth and screen size.

`CGDisplayBestModeForParametersAndRefreshRate` (page 26)
Returns information about the display mode closest to a specified depth, screen size, and refresh rate.

`CGDisplayBestModeForParametersAndRefreshRateWithProperty` (page 28)
Returns information about the display mode closest to a specified depth, screen size, and refresh rate, with a required property.

`CGDisplayCurrentMode` (page 32)
Returns information about the current display mode.

`CGDisplaySwitchToMode` (page 50)
Switches a display to a different mode.

Adjusting the Display Gamma

`CGSetDisplayTransferByFormula` (page 71)
Sets the gamma function for a display, by specifying the coefficients of the gamma transfer formula.

`CGGetDisplayTransferByFormula` (page 56)
Gets the coefficients of the gamma transfer formula for a display.

`CGSetDisplayTransferByTable` (page 73)
Sets the color gamma function for a display, by specifying the values in the RGB gamma tables.

`CGGetDisplayTransferByTable` (page 58)
Gets the values in the RGB gamma tables for a display.

`CGSetDisplayTransferByByteTable` (page 70)
Sets the byte values in the 8-bit RGB gamma tables for a display.

`CGDisplayRestoreColorSyncSettings` (page 46)
Restores the gamma tables to the values in the user's ColorSync display profile.

`CGDisplayGammaTableCapacity` (page 35)
Returns the capacity, or number of entries, in the gamma table for a display.

Working With Color Palettes

`CGPaletteCreateDefaultColorPalette` (page 61)
Returns a new display palette representing the default 8-bit color palette.

`CGPaletteCreateFromPaletteBlendedWithColor` (page 62)
Returns a new tinted display palette. The new palette is derived from an existing palette blended with a solid color, at a specified level of intensity.

`CGPaletteCreateWithByteSamples` (page 62)
Returns a new display palette using 8-bit sample data.

`CGPaletteCreateWithCapacity` (page 63)
Returns a new display palette with a specified capacity. The new palette is initialized from the default color palette.

`CGPaletteCreateWithDisplay` (page 63)
Returns a copy of the current palette for a display.

`CGPaletteCreateWithSamples` (page 63)
Returns a new display palette using RGB sample data.

`CGPaletteCreateCopy` (page 61)
Returns a copy of a specified display palette.

`CGPaletteRelease` (page 66)
Decrements the retain count of a display palette.

`CGPaletteGetColorAtIndex` (page 64)
Returns the color value at the specified index.

`CGPaletteGetIndexForColor` (page 64)
Returns the index of the display palette entry that most closely matches a specified color value.

`CGPaletteGetNumberOfSamples` (page 65)
Returns the number of colors in a display palette.

`CGPaletteIsEqualToPalette` (page 65)
Returns a Boolean value indicating whether two display palettes are equal.

`CGPaletteSetColorAtIndex` (page 66)
Updates the color value at the specified index in a display palette.

`CGDisplayCanSetPalette` (page 30)
Returns a Boolean value indicating whether the current display mode supports palettes.

`CGDisplaySetPalette` (page 48)
Sets the palette for a display.

Display Fade Effects

`CGConfigureDisplayFadeEffect` (page 18)
Modifies the settings of the built-in fade effect that occurs during a display configuration.

`CGAcquireDisplayFadeReservation` (page 14)
Reserves the fade hardware for a specified time interval.

`CGDisplayFade` (page 33)
Performs a single fade operation.

`CGDisplayFadeOperationInProgress` (page 34)
Returns a Boolean value indicating whether a fade operation is currently in progress.

`CGReleaseDisplayFadeReservation` (page 68)
Releases a display fade reservation, and unfades the display if needed.

Beam Position

These functions are advisory in nature and depend on IO Kit and hardware-specific drivers to implement support. If you need extremely precise timing, or access to vertical blanking interrupts, you should consider writing a device driver to tie into hardware-specific capabilities.

`CGDisplayBeamPosition` (page 25)
Returns the current beam position on a display.

`CGDisplayWaitForBeamPositionOutsideLines` (page 53)
Waits until the beam position moves outside a region in a display screen. This function is not designed for VBL drawing synchronization.

Controlling the Mouse Cursor

`CGDisplayHideCursor` (page 35)
Hides the mouse cursor, and increments the hide cursor count.

`CGDisplayShowCursor` (page 50)
Decrements the hide cursor count, and shows the mouse cursor if the count is zero.

`CGDisplayMoveCursorToPoint` (page 43)
Moves the mouse cursor to a specified point relative to the display origin (the upper left corner of the display).

`CGCursorIsVisible` (page 23)
Returns a Boolean value indicating whether the mouse cursor is visible.

`CGCursorIsDrawnInFramebuffer` (page 22)
Returns a Boolean value indicating whether the mouse cursor is drawn in frame buffer memory.

`CGAssociateMouseAndMouseCursorPosition` (page 15)
Connects or disconnects the mouse and cursor while an application is in the foreground.

`CGWarpMouseCursorPosition` (page 77)
Moves the mouse cursor without generating events.

`CGGetLastMouseDelta` (page 58)
Reports the change in mouse position since the last mouse movement event received by the application.

Getting Window Server Information

`CGSessionCopyCurrentDictionary` (page 70)
Returns information about the caller's window server session.

`CGWindowServerCFMachPort` (page 78)
: Returns a Core Foundation mach port (CFMachPort) that corresponds to the Mac OS X window server.

`CGWindowLevelForKey` (page 77)
: Returns the window level that corresponds to one of the standard window types.

Getting Information About Refresh and Move Operations

You can use these functions to find out what areas on local displays are changing their appearance as the result of operations such as drawing, window movement or scrolling, and display reconfiguration.

`CGRegisterScreenRefreshCallback` (page 67)
: Registers a callback function to be invoked when local displays are refreshed or modified.

`CGUnregisterScreenRefreshCallback` (page 74)
: Removes a previously registered callback function invoked when local displays are refreshed or modified.

`CGWaitForScreenRefreshRects` (page 75)
: Waits for screen refresh operations.

`CGScreenRegisterMoveCallback` (page 69)
: Registers a callback function to be invoked when an area of the display is moved.

`CGScreenUnregisterMoveCallback` (page 70)
: Removes a previously registered callback function invoked when an area of the display is moved.

`CGWaitForScreenUpdateRects` (page 76)
: Waits for screen update operations.

`CGReleaseScreenRefreshRects` (page 68)
: Deallocates a list of rectangles that represent changed areas on local displays.

Functions

CGAcquireDisplayFadeReservation

Reserves the fade hardware for a specified time interval.

```
CGError CGAcquireDisplayFadeReservation (
   CGDisplayReservationInterval seconds,
   CGDisplayFadeReservationToken *pNewToken
);
```

Parameters

seconds
: The desired number of seconds to reserve the fade hardware. An application can specify any value in the interval `(0, kCGMaxDisplayReservationInterval]`.

pNewToken
: A pointer to storage (provided by the caller) for a fade reservation token. On return, the storage contains a new token.

Return Value
Returns `kCGErrorNoneAvailable` if another fade reservation is in effect. Otherwise, returns `kCGErrorSuccess`.

Discussion
Before performing a fade operation, an application must reserve the fade hardware for a specified period of time. Quartz returns a token that represents a new fade reservation. The application uses this token as an argument in subsequent calls to other display fade functions.

During the fade reservation interval, the application has exclusive rights to use the fade hardware. At the end of the interval, the token becomes invalid and the hardware automatically returns to a normal state. Typically the application calls `CGReleaseDisplayFadeReservation` (page 68) to release the fade reservation before it expires.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGAssociateMouseAndMouseCursorPosition

Connects or disconnects the mouse and cursor while an application is in the foreground.

```
CGError CGAssociateMouseAndMouseCursorPosition (
   boolean_t connected
);
```

Parameters

connected
Pass `true` if the mouse and cursor should be connected; otherwise, pass `false`.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
When you call this function to disconnect the cursor and mouse, all events received by your application have a constant absolute location but contain mouse delta (change in X and Y) data. You may hide the cursor or change it into something appropriate for your application. You can reposition the cursor by using the function `CGDisplayMoveCursorToPoint` (page 43) or the function `CGWarpMouseCursorPosition` (page 77).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGRemoteOperation.h`

CGBeginDisplayConfiguration

Begins a new set of display configuration changes.

```
CGError CGBeginDisplayConfiguration (
   CGDisplayConfigRef *pConfigRef
);
```

Parameters

pConfigRef

A pointer to storage you provide for a display configuration. On return, your storage contains a new display configuration.

Return Value

A result code. If the object is successfully created, the result is `kCGErrorSuccess`. For other possible values, see "Quartz Display Services Result Codes" (page 102).

Discussion

This function creates a display configuration object that provides a context for a set of display configuration changes. After you specify the desired changes, you use `CGCompleteDisplayConfiguration` (page 17) to apply them in a single transaction.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGDisplayConfiguration.h`

CGCancelDisplayConfiguration

Cancels a set of display configuration changes.

```
CGError CGCancelDisplayConfiguration (
   CGDisplayConfigRef configRef
);
```

Parameters

configRef

The display configuration to cancel. On return, the configuration is cancelled and is no longer valid.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion

This function is used to abandon a display configuration. As a side effect, the display configuration object is released.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGDisplayConfiguration.h`

CGCaptureAllDisplays

Captures all attached displays.

```
CGDisplayErr CGCaptureAllDisplays (
   void
);
```

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
This function captures all attached displays in a single operation. This operation provides an immersive environment for your application, and it prevents other applications from trying to adjust to display changes.

Availability
Available in Mac OS X v10.0 and later.

See Also
`CGDisplayCapture` (page 31)

Declared In
`CGDirectDisplay.h`

CGCaptureAllDisplaysWithOptions

Captures all attached displays, using the specified options.

```
CGDisplayErr CGCaptureAllDisplaysWithOptions (
   CGCaptureOptions options
);
```

Parameters

options
The options to use. See "Display Capture Options" (page 91).

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
This function allows you to specify one or more options to use during capture of all attached displays.

Availability
Available in Mac OS X v10.3 and later.

See Also
`CGCaptureAllDisplays` (page 16)

Declared In
`CGDirectDisplay.h`

CGCompleteDisplayConfiguration

Completes a set of display configuration changes.

```
CGError CGCompleteDisplayConfiguration (
   CGDisplayConfigRef configRef,
   CGConfigureOption option
);
```

Parameters

configRef

The display configuration with the desired changes. On return, this configuration is no longer valid.

option

The scope of the display configuration changes. Pass one of the constants listed in "Display Configuration Scopes" (page 93).

Return Value

A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion

This function applies a set of display configuration changes as a single atomic transaction. The duration or scope of the changes depends on the value of the `option` parameter. The possible scopes are fully described in "Display Configuration Scopes" (page 93).

A configuration change may fail if an unsupported display mode is requested, or if another application is running in full-screen mode.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGDisplayConfiguration.h`

CGConfigureDisplayFadeEffect

Modifies the settings of the built-in fade effect that occurs during a display configuration.

```
CGError CGConfigureDisplayFadeEffect (
   CGDisplayConfigRef configRef,
   CGDisplayFadeInterval fadeOutSeconds,
   CGDisplayFadeInterval fadeInSeconds,
   float fadeRed,
   float fadeGreen,
   float fadeBlue
);
```

Parameters

configRef

A display configuration, acquired by calling `CGBeginDisplayConfiguration` (page 15).

fadeOutSeconds

The time in seconds to fade from the normal display to the specified fade color. The fade out is completed before the display configuration is changed. If the interval is 0, Quartz applies the color immediately.

fadeInSeconds

Time in seconds to return from the specified fade color to the normal display. The fade-in is run asynchronously after the display configuration is changed.

fadeRed
An intensity value in the interval [0, 1] that represents the red component of the desired blend color.

fadeGreen
An intensity value in the interval [0, 1] that represents the green component of the desired blend color.

fadeBlue
An intensity value in the interval [0, 1] that represents the blue component of the desired blend color.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
This function provides a way to customize the built-in fade effect that Quartz performs when displays are reconfigured. The default time settings for this fade effect are 0.3 seconds to fade out, and 0.5 seconds to fade back in. The default fade color is French Blue for a normal desktop, and black for a captured display.

Before using this function, you need to call `CGBeginDisplayConfiguration` (page 15) to acquire the display configuration token for the desired display. No fade reservation is needed—when you call `CGCompleteDisplayConfiguration` (page 17), Quartz reserves the fade hardware (assuming it is available) and performs the fade.

Calling this function modifies the fade behavior for a single display configuration, and has no permanent effect.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGConfigureDisplayMirrorOfDisplay

Changes the configuration of a mirroring set.

```
CGError CGConfigureDisplayMirrorOfDisplay (
   CGDisplayConfigRef configRef,
   CGDirectDisplayID display,
   CGDirectDisplayID masterDisplay
);
```

Parameters

configRef
A display configuration, acquired by calling `CGBeginDisplayConfiguration` (page 15).

display
The display to add to a mirroring set.

masterDisplay
A display in a mirroring set, or `kCGNullDirectDisplay` to disable mirroring. To specify the main display, use `CGMainDisplayID` (page 60).

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
Display mirroring and display matte generation are implemented either in hardware (preferred) or software, at the discretion of the device driver.

- Hardware mirroring

  With hardware mirroring enabled, all drawing is directed to the primary display—see `CGDisplayPrimaryDisplay` (page 44).

  If the device driver selects hardware matte generation, the display bounds and rowbytes values are adjusted to reflect the active drawable area.

- Software mirroring

  In this form of mirroring, identical content is drawn into each display in the mirroring set. Applications that use the window system need not be concerned about mirroring, as the window system takes care of all flushing of window content to the appropriate displays.

  Applications that draw directly to the display, as with display capture, must make sure to draw the same content to all mirrored displays in a software mirror set. When drawing to software mirrored displays using a full screen OpenGL context (not drawing through a window), you should create shared OpenGL contexts for each display and re-render for each display.

You can use the function `CGGetActiveDisplayList` (page 53) to determine which displays are active, or drawable. This automatically gives your application the correct view of the current displays.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGConfigureDisplayMode

Configures the display mode of a display.

```
CGError CGConfigureDisplayMode (
   CGDisplayConfigRef configRef,
   CGDirectDisplayID display,
   CFDictionaryRef mode
);
```

Parameters

configRef
A display configuration, acquired by calling `CGBeginDisplayConfiguration` (page 15).

display
The display being configured.

mode
A display mode dictionary (see the discussion below).

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
A display mode is a set of properties such as width, height, pixel depth, and refresh rate, and options such as stretched LCD panel filling.

The display mode you provide must be one of the following:

- A dictionary returned by one of the `CGDisplayBestMode` functions, such as `CGDisplayBestModeForParameters` (page 26).
- A dictionary in the array returned by `CGDisplayAvailableModes` (page 24).

If you use this function to change the mode of a display in a mirroring set, Quartz may adjust the bounds, resolutions, and depth of the other displays in the set to a safe mode, with matching depth and the smallest enclosing size.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGConfigureDisplayOrigin

Configures the origin of a display in global display (desktop) coordinates.

```
CGError CGConfigureDisplayOrigin (
   CGDisplayConfigRef configRef,
   CGDirectDisplayID display,
   CGDisplayCoord x,
   CGDisplayCoord y
);
```

Parameters

configRef
A display configuration, acquired by calling `CGBeginDisplayConfiguration` (page 15).

display
The display being configured.

x
The desired x-coordinate for the upper left corner of the display.

y
The desired y-coordinate for the upper left corner of the display.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
In Quartz, the upper left corner of a display is called the origin. The origin of a display is always specified in global display (desktop) coordinates. The origin of the main or primary display is (0,0).

The new origin is placed as close as possible to the requested location, without overlapping or leaving a gap between displays.

If you use this function to change the origin of a mirrored display, the display may be removed from the mirroring set.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGConfigureDisplayStereoOperation

Enables or disables stereo operation for a display, as part of a display configuration.

```
CGError CGConfigureDisplayStereoOperation (
   CGDisplayConfigRef configRef,
   CGDirectDisplayID display,
   boolean_t stereo,
   boolean_t forceBlueLine
);
```

Parameters

configRef
: A display configuration, acquired by calling `CGBeginDisplayConfiguration` (page 15).

display
: The display being configured.

stereo
: Pass `true` if you want to enable stereo operation. To disable it, pass `false`.

forceBlueLine
: When in stereo operation, a display may need to generate a special stereo sync signal as part of the video output. The sync signal consists of a blue line which occupies the first 25% of the last scanline for the left eye view, and the first 75% of the last scanline for the right eye view. The remainder of the scanline is black. To force the display to generate this sync signal, pass `true`; otherwise, pass `false`.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
The system normally detects the presence of a stereo window and automatically switches a display containing a stereo window to stereo operation. This function provides a mechanism to force a display to stereo operation, and to set options (blue line sync signal) when in stereo operation.

On success, the display resolution, mirroring mode, and available display modes may change due to hardware-specific capabilities and limitations. You should check these settings to verify that they are appropriate for your application.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGDisplayConfiguration.h`

CGCursorIsDrawnInFramebuffer

Returns a Boolean value indicating whether the mouse cursor is drawn in frame buffer memory.

```
boolean_t CGCursorIsDrawnInFramebuffer (
   void
);
```

Return Value
If `true`, the cursor is drawn in frame buffer memory; otherwise, `false`.

Discussion
This function returns a Boolean value that indicates whether or not the cursor is drawn in the frame buffer. (The cursor could exist in an overlay plane or a similar mechanism that puts pixels on-screen without altering frame buffer content.) If the cursor is drawn in the frame buffer, it is read back along with window data.

The reported Boolean value is based on the union of the state of the cursor on all displays. If the cursor is drawn in the frame buffer on any display, the function returns `true`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGRemoteOperation.h`

CGCursorIsVisible

Returns a Boolean value indicating whether the mouse cursor is visible.

```
boolean_t CGCursorIsVisible (
   void
);
```

Return Value
If `true`, the cursor is visible on any display; otherwise, `false`.

Discussion
To hide or show the cursor, you can use the functions `CGDisplayHideCursor` (page 35) and `CGDisplayShowCursor` (page 50).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGRemoteOperation.h`

CGDisplayAddressForPosition

Returns the address in frame buffer memory that corresponds to a position on an online display.

```
void * CGDisplayAddressForPosition (
   CGDirectDisplayID display,
   CGDisplayCoord x,
   CGDisplayCoord y
);
```

Parameters

display

The display to access.

x

The x-coordinate of a position in global display space. The origin is the upper left corner of the main display.

y

The y-coordinate of a position in global display space. The origin is the upper left corner of the main display, and the y-axis is oriented down.

Return Value

The address in frame buffer memory that corresponds to the specified position. If the display ID is invalid or the point lies outside the bounds of the display, the return value is `NULL`.

Discussion

If the display has not been captured, the returned address may refer to read-only memory.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayAvailableModes

Returns information about the currently available display modes.

```
CFArrayRef CGDisplayAvailableModes (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

An array of dictionaries with display mode information, or `NULL` if the display is invalid. The array is owned by the system and you should not release it. Each dictionary in the array contains information about a mode that the display supports. For a list of the properties in a display mode dictionary, see "Display Mode Standard Properties" (page 95) and "Display Mode Optional Properties" (page 96). For general information about using dictionaries, see *CFDictionary Reference*.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

QTCarbonShell

Declared In
`CGDirectDisplay.h`

CGDisplayBaseAddress

Returns the base address in frame buffer memory of an online display.

```
void * CGDisplayBaseAddress (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The base address in frame buffer memory of the specified display. If the display ID is invalid, the return value is `NULL`.

Discussion
If the display has not been captured, the returned address may refer to read-only memory.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayBeamPosition

Returns the current beam position on a display.

```
CGBeamPosition CGDisplayBeamPosition (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The current beam position on the specified display. If the display does not implement conventional video vertical and horizontal sweep in painting, or the driver does not implement this functionality, 0 is returned.

Discussion
This function returns the number of the scan line on which the beam is currently positioned, expressed as a non-negative integer. The value increases as the beam moves lower on the display.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayBestModeForParameters

Returns information about the display mode closest to a specified depth and screen size.

```
CFDictionaryRef CGDisplayBestModeForParameters (
   CGDirectDisplayID display,
   size_t bitsPerPixel,
   size_t width,
   size_t height,
   boolean_t *exactMatch
);
```

Parameters

display

The display to optimize.

bitsPerPixel

Optimal display depth in bits per pixel. Note that this value is not the same as pixel depth, which is the number of bits per channel or component.

width

Optimal display width in pixel units.

height

Optimal display height in pixel units.

exactMatch

A pointer to a Boolean variable. On return, its value is `true` if an exact match in display depth, width, and height is found; otherwise, `false`. If this information is not needed, pass `NULL`.

Return Value

A display mode dictionary, or `NULL` if the display is invalid. The dictionary is owned by the system and you should not release it. The dictionary contains information about the display mode closest to the specified depth and screen size. For a list of the properties in a display mode dictionary, see "Display Mode Standard Properties" (page 95) and "Display Mode Optional Properties" (page 96). For general information about using dictionaries, see *CFDictionary Reference*.

Discussion

This function tries to find an optimal display mode for the specified display. The function first tries to find a mode with the specified pixel depth and dimensions equal to or greater than the specified width and height. If no depth match is found, it tries to find a mode with greater depth and the same or greater dimensions. If a suitable display mode is not found, this function simply returns the current display mode.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayBestModeForParametersAndRefreshRate

Returns information about the display mode closest to a specified depth, screen size, and refresh rate.

```
CFDictionaryRef CGDisplayBestModeForParametersAndRefreshRate (
   CGDirectDisplayID display,
   size_t bitsPerPixel,
   size_t width,
   size_t height,
   CGRefreshRate refresh,
   boolean_t *exactMatch
);
```

Parameters

display

The display to access.

bitsPerPixel

Optimal display depth, in bits per pixel. Note that this value is not the same as pixel depth, which is the number of bits per channel or component.

width

Optimal display width, in pixel units.

height

Optimal display height, in pixel units.

refresh

Optimal display refresh rate, in frames per second.

exactMatch

A pointer to a Boolean variable. On return, its value is `true` if an exact match in display depth, width, height, and refresh rate is found; otherwise, `false`. If this information is not needed, pass `NULL`.

Return Value

A display mode dictionary, or `NULL` if the display is invalid. The dictionary is owned by the system and you should not release it. The dictionary contains information about the display mode closest to the specified depth, screen size, and refresh rate. For a list of the properties in a display mode dictionary, see "Display Mode Standard Properties" (page 95) and "Display Mode Optional Properties" (page 96). For general information about using dictionaries, see *CFDictionary Reference*.

Discussion

This function searches the list of available display modes for a mode that comes closest to satisfying these criteria:

- Has a pixel depth equal to or greater than the specified depth
- Has dimensions equal to or greater than the specified height and width
- Uses a refresh rate equal to or near the specified rate

If a suitable display mode is not found, this function simply returns the current display mode.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayBestModeForParametersAndRefreshRateWithProperty

Returns information about the display mode closest to a specified depth, screen size, and refresh rate, with a required property.

```
CFDictionaryRef CGDisplayBestModeForParametersAndRefreshRateWithProperty (
   CGDirectDisplayID display,
   size_t bitsPerPixel,
   size_t width,
   size_t height,
   CGRefreshRate refresh,
   CFStringRef property,
   boolean_t *exactMatch
);
```

Parameters

display

The display to access.

bitsPerPixel

Optimal display depth, in bits per pixel. Note that this value is not the same as pixel depth, which is the number of bits per channel or component.

width

Optimal display width, in pixels.

height

Optimal display height, in pixels.

refresh

Optimal display refresh rate, in refreshes per second.

property

A required display mode property. For a list of the properties you can specify, see "Display Mode Optional Properties" (page 96).

exactMatch

A pointer to a Boolean variable. On return, its value is `true` if an exact match in display depth, width, height, refresh rate, and property is found; otherwise, `false`. If this information is not needed, pass `NULL`.

Return Value

A display mode dictionary, or `NULL` if the display is invalid. The dictionary is owned by the system and you should not release it. The dictionary contains information about the display mode with the specified property that comes closest to the specified depth, screen size, and refresh rate. For a list of the properties in a display mode dictionary, see "Display Mode Standard Properties" (page 95) and "Display Mode Optional Properties" (page 96). For general information about using dictionaries, see *CFDictionary Reference*.

Discussion

This function searches the list of available display modes for a mode that includes the specified property and comes closest to satisfying these criteria:

- Has a pixel depth equal to or greater than the specified depth
- Has dimensions equal to or greater than the specified height and width
- Uses a refresh rate equal to or near the specified rate

If no matching display mode is found, this function simply returns the current display mode.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayBitsPerPixel

Returns the number of bits used to represent a pixel in the frame buffer.

```
size_t CGDisplayBitsPerPixel (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The number of bits used to represent a pixel in the frame buffer.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayBitsPerSample

Returns the number of bits used to represent a pixel component in the frame buffer.

```
size_t CGDisplayBitsPerSample (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The number of bits used to represent a pixel component such as a color value in the frame buffer.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayBounds

Returns the bounds of a display in global display space.

```
CGRect CGDisplayBounds (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

The bounds of the display, expressed as a rectangle in the global display coordinate space (relative to the upper left corner of the main display).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayBytesPerRow

Returns the number of bytes per row in a display.

```
size_t CGDisplayBytesPerRow (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

The number of bytes per row in the display. This number also represents the stride between pixels in the same column of the display.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayCanSetPalette

Returns a Boolean value indicating whether the current display mode supports palettes.

```
boolean_t CGDisplayCanSetPalette (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

If `true`, the current display mode supports palettes; otherwise, `false`.

Discussion
Palettes are supported in any display selected to run in a 256-color display mode.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayCapture

Captures a display for exclusive use by an application.

```
CGDisplayErr CGDisplayCapture (
   CGDirectDisplayID display
);
```

Parameters

display
The display to capture.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
When an application captures a display, Quartz does not allow other applications and system services to use the display or change its configuration.

If hardware or software mirroring is in effect, the easiest way to capture the primary display and all mirrored displays is to use the function `CGCaptureAllDisplays` (page 16). In case of software mirroring, applications that draw directly to the display must make sure to draw the same content to all displays in the mirror set.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayCaptureWithOptions

Captures a display for exclusive use by an application, using the specified options.

```
CGDisplayErr CGDisplayCaptureWithOptions (
   CGDirectDisplayID display,
   CGCaptureOptions options
);
```

Parameters

display
The display to capture.

options
The options to use. See "Display Capture Options" (page 91).

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
This function allows you to specify one or more options to use during capture of a display.

Availability
Available in Mac OS X v10.3 and later.

See Also
`CGDisplayCapture` (page 31)

Declared In
`CGDirectDisplay.h`

CGDisplayCopyColorSpace

Returns the color space for a display.

```
CGColorSpaceRef CGDisplayCopyColorSpace (
    CGDirectDisplayID display
);
```

Parameters

display
The display whose color space you want to obtain.

Return Value
The current color space for the specified display. The caller is responsible for releasing the color space with the `CGColorSpaceRelease` function.

Discussion
This function returns a display-dependent ICC-based color space. You can use this function when rendering content for a specific display in order to produce color-matched output for that display.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayCurrentMode

Returns information about the current display mode.

```
CFDictionaryRef CGDisplayCurrentMode (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
A display mode dictionary, or `NULL` if the display is invalid. The dictionary is owned by the system and you should not release it. The dictionary contains information about the current display mode. For a list of the properties in a display mode dictionary, see "Display Mode Standard Properties" (page 95) and "Display Mode Optional Properties" (page 96). For general information about using dictionaries, see *CFDictionary Reference*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayFade

Performs a single fade operation.

```
CGError CGDisplayFade (
   CGDisplayFadeReservationToken myToken,
   CGDisplayFadeInterval seconds,
   CGDisplayBlendFraction startBlend,
   CGDisplayBlendFraction endBlend,
   float redBlend,
   float greenBlend,
   float blueBlend,
   boolean_t synchronous
);
```

Parameters

myToken
A reservation token for the fade hardware, acquired by calling `CGAcquireDisplayFadeReservation` (page 14).

seconds
The desired number of seconds for the fade operation. You should use a value in the interval `[0, kCGMaxDisplayReservationInterval]`. If the value is 0, the ending blend color is applied immediately.

startBlend
An intensity value in the interval [0, 1] that specifies the alpha component of the desired blend color at the beginning of the fade operation. See "Display Fade Blend Fractions" (page 94).

endBlend
An intensity value in the interval [0, 1] that specifies the alpha component of the desired blend color at the end of the fade operation. See "Display Fade Blend Fractions" (page 94).

redBlend
An intensity value in the interval [0, 1] that specifies the red component of the desired blend color.

greenBlend
An intensity value in the interval [0, 1] that specifies the green component of the desired blend color.

blueBlend
An intensity value in the interval [0, 1] that specifies the blue component of the desired blend color.

synchronous

Pass `true` if you want the fade operation to be synchronous; otherwise, pass `false`. If a fade operation is synchronous, the function does not return until the operation is complete.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
Over the fade operation time interval, Quartz interpolates a blending coefficient between the starting and ending values given, applying a nonlinear (sine-based) bias term. Using this coefficient, the video output is blended with the specified color.

The following example shows how to perform a two-second synchronous fade-out to black:

```
CGDisplayFade (
    myToken,
    2.0,                        // 2 seconds
    kCGDisplayBlendNormal,      // starting state
    kCGDisplayBlendSolidColor,  // ending state
    0.0, 0.0, 0.0,              // black
    true                        // wait for completion
);
```

To perform a two-second asynchronous fade-in from black:

```
CGDisplayFade (
    myToken,
    2.0,                        // 2 seconds
    kCGDisplayBlendSolidColor,  // starting state
    kCGDisplayBlendNormal,      // ending state
    0.0, 0.0, 0.0,              // black
    false                       // don't wait for completion
);
```

If you specify an asynchronous fade operation, it's safe to call `CGReleaseDisplayFadeReservation` (page 68) immediately after this function returns.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGDisplayFadeOperationInProgress

Returns a Boolean value indicating whether a fade operation is currently in progress.

```
boolean_t CGDisplayFadeOperationInProgress (
   void
);
```

Return Value
If `true`, a fade operation is currently in progress; otherwise, `false`.

Discussion
You may call this function from any task running on the system. The calling task need not have a valid fade reservation.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGDisplayGammaTableCapacity

Returns the capacity, or number of entries, in the gamma table for a display.

```
CGTableCount CGDisplayGammaTableCapacity (
   CGDirectDisplayID display
);
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayGetDrawingContext

Returns a graphics context suitable for drawing to a captured display.

```
CGContextRef CGDisplayGetDrawingContext (
   CGDirectDisplayID display
);
```

Parameters

`display`
The display to access.

Return Value
A Quartz graphics context suitable for drawing to a captured display, or `NULL` if the display has not been captured. The context is owned by the system and you should not release it.

Discussion
After capturing a display or changing the configuration of a captured display, you can use this function to obtain the current graphics context for the display. The graphics context remains valid while the display is captured and the display configuration is unchanged. Releasing the captured display or reconfiguring the display invalidates the context. To determine when the display configuration is changing, you can use the function `CGDisplayRegisterReconfigurationCallback` (page 45) to register a display reconfiguration callback.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayHideCursor

Hides the mouse cursor, and increments the hide cursor count.

```
CGDisplayErr CGDisplayHideCursor (
   CGDirectDisplayID display
);
```

Parameters

display

This parameter is not used. By default, you may pass `kCGDirectMainDisplay`.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion

This function hides the cursor regardless of its current location; the `display` parameter is ignored. In most cases, the caller must be the foreground application to affect the cursor.

Availability

Available in Mac OS X v10.0 and later.

See Also

`CGDisplayShowCursor` (page 50)

Declared In

`CGDirectDisplay.h`

CGDisplayIDToOpenGLDisplayMask

Maps a display ID to an OpenGL display mask.

```
CGOpenGLDisplayMask CGDisplayIDToOpenGLDisplayMask (
   CGDirectDisplayID display
);
```

Parameters

display

The display ID to be converted.

Return Value

The OpenGL display mask that corresponds to the specified display.

Discussion

OpenGL sometimes identifies a display using a bitmask with one bit set. This function maps a display ID to the corresponding OpenGL display mask.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayIOServicePort

Returns the I/O Kit service port of the specified display.

```
io_service_t CGDisplayIOServicePort (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value
The I/O Kit service port for the specified display.

Discussion
An I/O Kit service port can be passed to I/O Kit to obtain additional information about the display.

The port is owned by the graphics system, and should not be destroyed.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayIsActive

Returns a Boolean value indicating whether a display is active.

```
boolean_t CGDisplayIsActive (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value
If `true`, the specified display is active; otherwise, `false`.

Discussion
An active display is connected, awake, and available for drawing. In a hardware mirroring set, only the primary display is active.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayIsAlwaysInMirrorSet

Returns a Boolean value indicating whether a display is always in a mirroring set.

```
boolean_t CGDisplayIsAlwaysInMirrorSet (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

If `true`, the specified display is in a mirroring set and cannot be removed from this set.

Discussion

Some hardware configurations support the connection of auxiliary displays that always mirror the main display, and therefore cannot be removed from the mirroring set to which they belong.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGDisplayConfiguration.h`

CGDisplayIsAsleep

Returns a Boolean value indicating whether a display is sleeping (and is therefore not drawable.)

```
boolean_t CGDisplayIsAsleep (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

If `true`, the specified display is in sleep mode; otherwise, `false`.

Discussion

A display is sleeping when its frame buffer and the attached monitor are in reduced power mode. A sleeping display is still considered to be a part of global display (desktop) space, but it is not drawable.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGDisplayConfiguration.h`

CGDisplayIsBuiltin

Returns a Boolean value indicating whether a display is built-in, such as the internal display in portable systems.

```
boolean_t CGDisplayIsBuiltin (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

If `true`, the specified display is considered to be a built-in display; otherwise, `false`.

Discussion

Portable systems typically identify the internal LCD panel as a built-in display.

Note that it is possible and reasonable for a system to have no displays marked as built-in. For example, a portable system running with the lid closed may report no built-in displays.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGDisplayConfiguration.h`

CGDisplayIsCaptured

Returns a Boolean value indicating whether a display is captured.

```
boolean_t CGDisplayIsCaptured (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

If `true`, the specified display is captured; otherwise, `false`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayIsInHWMirrorSet

Returns a Boolean value indicating whether a display is in a hardware mirroring set.

```
boolean_t CGDisplayIsInHWMirrorSet (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value
If `true`, the specified display is a member of a hardware mirroring set; otherwise, `false`.

Discussion
When hardware mirroring is enabled, the contents of a single frame buffer are rendered in all displays in the hardware mirroring set. All drawing operations are directed to the primary display in the set—see `CGDisplayPrimaryDisplay` (page 44).

For more information about display mirroring, see `CGConfigureDisplayMirrorOfDisplay` (page 19).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayIsInMirrorSet

Returns a Boolean value indicating whether a display is in a mirroring set.

```
boolean_t CGDisplayIsInMirrorSet (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
If `true`, the specified display is a member of a software or hardware mirroring set; otherwise, `false`.

Discussion
For more information about display mirroring, see `CGConfigureDisplayMirrorOfDisplay` (page 19).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayIsMain

Returns a Boolean value indicating whether a display is the main display.

```
boolean_t CGDisplayIsMain (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
If `true`, the specified display is currently the main display; otherwise, `false`.

Discussion
For information about the characteristics of a main display, see `CGMainDisplayID` (page 60).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayIsOnline

Returns a Boolean value indicating whether a display is connected or online.

```
boolean_t CGDisplayIsOnline (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
If `true`, the specified display is connected; otherwise, `false`.

Discussion
A display is considered connected or online when the frame buffer hardware is connected to a monitor.

You can use this function to determine if someone has hot-plugged a display to the system. Note that hot-plugging is a hardware feature that may not be present on all displays.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayIsStereo

Returns a Boolean value indicating whether a display is running in a stereo graphics mode.

```
boolean_t CGDisplayIsStereo (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
If `true`, the specified display is running in a stereo graphics mode; otherwise, `false`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayMirrorsDisplay

For a secondary display in a mirroring set, returns the primary display.

```
CGDirectDisplayID CGDisplayMirrorsDisplay (
   CGDirectDisplayID display
);
```

Parameters

display
A secondary display in a mirroring set.

Return Value
Returns the primary display in the mirroring set. Returns `kCGNullDirectDisplay` if the specified display is actually the primary display or is not in a mirroring set.

Discussion
For more information about display mirroring, see `CGConfigureDisplayMirrorOfDisplay` (page 19).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayModelNumber

Returns the model number of a display monitor.

```
uint32_t CGDisplayModelNumber (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
A model number for the monitor associated with the specified display, or a constant to indicate an exception—see the discussion below.

Discussion
This function uses I/O Kit to identify the monitor associated with the specified display. The return value depends on the following:

- If I/O Kit can identify the monitor, the product ID code for the monitor is returned.
- If I/O Kit can't identify the monitor, `kDisplayProductIDGeneric` is returned.
- If no monitor is connected, a value of `0xFFFFFFFF` is returned.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayMoveCursorToPoint

Moves the mouse cursor to a specified point relative to the display origin (the upper left corner of the display).

```
CGDisplayErr CGDisplayMoveCursorToPoint (
   CGDirectDisplayID display,
   CGPoint point
);
```

Parameters

display
The display to access.

point
The coordinates of a point in local display space. The origin is the upper left corner of the specified display.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
No events are generated as a result of this move. Points that would lie outside the desktop are clipped to the desktop.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayPixelsHigh

Returns the display height in pixel units.

```
size_t CGDisplayPixelsHigh (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The display height in pixel units.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayPixelsWide

Returns the display width in pixel units.

```
size_t CGDisplayPixelsWide (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The display width in pixel units.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayPrimaryDisplay

Returns the primary display in a hardware mirroring set.

```
CGDirectDisplayID CGDisplayPrimaryDisplay (
   CGDirectDisplayID display
);
```

Parameters

display
A display in a hardware mirror set.

Return Value
The primary display in the mirror set. If `display` is not hardware-mirrored, this function simply returns `display`.

Discussion
In hardware mirroring, the contents of a single frame buffer are rendered in two or more displays simultaneously. The mirrored displays are said to be in a *hardware mirroring set*.

At the discretion of the device driver, one of the displays in a hardware mirroring set is designated as the *primary* display. The device driver binds the drawing engine, hardware accelerator, and 3D engine to the primary display, and directs all drawing operations to this display.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayRegisterReconfigurationCallback

Registers a callback function to be invoked whenever a local display is reconfigured.

```
CGError CGDisplayRegisterReconfigurationCallback (
   CGDisplayReconfigurationCallBack proc,
   void *userInfo
);
```

Parameters

proc

A pointer to the callback function to be registered.

userInfo

A pointer to user-defined data, or `NULL`. The `userInfo` argument is passed back to the callback function each time it's invoked.

Discussion

Whenever local displays are reconfigured, the callback function you register is invoked twice for each display that's added, removed, or currently online—once before the reconfiguration, and once after the reconfiguration. For more information, see the callback type `CGDisplayReconfigurationCallBack` (page 78).

A callback function may be registered multiple times with different user-defined data pointers, resulting in multiple registration entries. For each registration, when notification is no longer needed you should remove the registration by calling the function `CGDisplayRemoveReconfigurationCallback` (page 46).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGDisplayConfiguration.h`

CGDisplayRelease

Releases a captured display.

```
CGDisplayErr CGDisplayRelease (
   CGDirectDisplayID display
);
```

Parameters

display

The display to release.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 102).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayRemoveReconfigurationCallback

Removes the registration of a callback function that's invoked whenever a local display is reconfigured.

```
CGError CGDisplayRemoveReconfigurationCallback (
   CGDisplayReconfigurationCallBack proc,
   void *userInfo
);
```

Parameters

proc

A pointer to the callback function associated with the registration to be removed.

userInfo

A pointer to user-defined data associated with the registration to be removed, or `NULL`. This is the same pointer that's passed to the function `CGDisplayRegisterReconfigurationCallback` (page 45) when registering the callback.

Discussion

When you call this function, the two arguments must match the registered entry to be removed.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGDisplayConfiguration.h`

CGDisplayRestoreColorSyncSettings

Restores the gamma tables to the values in the user's ColorSync display profile.

```
void CGDisplayRestoreColorSyncSettings (
   void
);
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayRotation

Returns the rotation angle of a display in degrees.

```
double CGDisplayRotation (
    CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

The rotation angle of the display in degrees, or 0 if the display is not valid.

Discussion
This function returns the rotation angle of a display in a clockwise direction. For example, if the specified display is rotated clockwise 90 degrees then this function returns 90.0. After a 90 degree clockwise rotation, the physical bottom of the display is on the left side and the physical top is on the right side.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplaySamplesPerPixel

Returns the number of color components used to represent a pixel.

```
size_t CGDisplaySamplesPerPixel (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The number of color components used to represent a pixel.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayScreenSize

Returns the width and height of a display in millimeters.

```
CGSize CGDisplayScreenSize (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The size of the specified display in millimeters, or 0 if the display is not valid.

Discussion
If Extended Display Identification Data (EDID) for the display device is not available, the size is estimated based on the device width and height in pixels from `CGDisplayBounds` (page 29), with an assumed resolution of 2.835 pixels/mm or 72 DPI, a reasonable guess for displays predating EDID support.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplaySerialNumber

Returns the serial number of a display monitor.

```
uint32_t CGDisplaySerialNumber (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
A serial number for the monitor associated with the specified display, or a constant to indicate an exception—see the discussion below.

Discussion
This function uses I/O Kit to identify the monitor associated with the specified display.

If I/O Kit can identify the monitor:

- If the manufacturer has encoded a serial number for the monitor, the number is returned.
- If there is no encoded serial number, `0x00000000` is returned.

If I/O Kit cannot identify the monitor:

- If a monitor is connected to the display, `0x00000000` is returned.
- If no monitor is connected to the display hardware, a value of `0xFFFFFFFF` is returned.

Note that a serial number is meaningful only in conjunction with a specific vendor and product or model.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplaySetPalette

Sets the palette for a display.

```
CGDisplayErr CGDisplaySetPalette (
   CGDirectDisplayID display,
   const CGDirectPaletteRef palette
);
```

Parameters

display
The display to access.

palette

The display palette to set.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplaySetStereoOperation

Immediately enables or disables stereo operation for a display.

```
CGError CGDisplaySetStereoOperation (
   CGDirectDisplayID display,
   boolean_t stereo,
   boolean_t forceBlueLine,
   CGConfigureOption option
);
```

Parameters

display

The display being configured.

stereo

Pass `true` if you want to enable stereo operation. To disable it, pass `false`.

forceBlueLine

When in stereo operation, a display may need to generate a special stereo sync signal as part of the video output. The sync signal consists of a blue line which occupies the first 25% of the last scanline for the left eye view, and the first 75% of the last scanline for the right eye view. The remainder of the scanline is black. To force the display to generate this sync signal, pass `true`; otherwise pass `false`.

option

A constant that specifies the scope of the display configuration changes. For more information, see "Display Configuration Scopes" (page 93).

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
The system normally detects the presence of a stereo window and automatically switches a display containing a stereo window to stereo operation. This function provides a mechanism to force a display to stereo operation immediately, and to set options (blue line sync signal) when in stereo operation.

On success, the display resolution, mirroring mode, and available display modes may change due to hardware-specific capabilities and limitations. You should check these settings to verify that they are appropriate for your application.

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGDisplayConfiguration.h

CGDisplayShowCursor

Decrements the hide cursor count, and shows the mouse cursor if the count is zero.

```
CGDisplayErr CGDisplayShowCursor (
   CGDirectDisplayID display
);
```

Parameters

display

This parameter is not used. By default, you may pass kCGDirectMainDisplay.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
If the hide cursor count is zero, this function shows the cursor regardless of its current location; the display parameter is ignored. In most cases, the caller must be the foreground application to affect the cursor.

Availability
Available in Mac OS X v10.0 and later.

See Also
CGDisplayHideCursor (page 35)

Declared In
CGDirectDisplay.h

CGDisplaySwitchToMode

Switches a display to a different mode.

```
CGDisplayErr CGDisplaySwitchToMode (
   CGDirectDisplayID display,
   CFDictionaryRef mode
);
```

Parameters

display

The display to access.

mode

A display mode dictionary that contains information about the display mode to set. The dictionary passed in must be a dictionary returned by another Quartz display function such as CGDisplayAvailableModes (page 24) or CGDisplayBestModeForParameters (page 26). For a list of the properties in a display mode dictionary, see "Display Mode Standard Properties" (page 95) and "Display Mode Optional Properties" (page 96). For general information about using dictionaries, see *CFDictionary Reference*.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
This function switches the display mode of the specified display. The operation is always synchronous; the function does not return until the mode switch is complete. Note that after switching, display parameters and addresses may change.

The selected display mode persists for the life of the calling program. When the program terminates, the display mode automatically reverts to the permanent setting in the Displays panel of System Preferences.

When changing the display mode of a display in a mirroring set, other displays in the mirroring set will be assigned a mode that's capable of mirroring the bounds of the display being adjusted. To avoid this automatic behavior, you can use the following procedure: call `CGBeginDisplayConfiguration`, call `CGConfigureDisplayMode` for each display to explicitly set the mode, and finally call `CGCompleteDisplayConfiguration`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayUnitNumber

Returns the logical unit number of a display.

```
uint32_t CGDisplayUnitNumber (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
A logical unit number for the specified display.

Discussion
The logical unit number represents a particular node in the I/O Kit device tree associated with the display's frame buffer. For a particular hardware configuration, this value will not change when the attached monitor is changed.

The unit number will change if the I/O Kit device tree changes, as when hardware is reconfigured, drivers are replaced, or significant changes occur to I/O Kit, so it should not be assumed to be invariant across login sessions.

For more information about I/O Kit, see the Apple publication *"I/O Kit Fundamentals"*.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayUsesOpenGLAcceleration

Returns a Boolean value indicating whether Quartz is using OpenGL-based window acceleration (Quartz Extreme) to render in a display.

```
boolean_t CGDisplayUsesOpenGLAcceleration (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

If `true`, Quartz Extreme is used to render in the specified display; otherwise, `false`.

Discussion

Quartz Extreme is an OpenGL-based, hardware-accelerated window compositor available in Mac OS X version 10.2 and later. Quartz Extreme requires a minimum hardware configuration to operate.

The information this function provides is typically used to adjust the demands of drawing operations to the capabilities of the display hardware. For example, an application running on an unaccelerated system could disable live window-resizing.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGDisplayConfiguration.h`

CGDisplayVendorNumber

Returns the vendor number of the specified display's monitor.

```
uint32_t CGDisplayVendorNumber (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

A vendor number for the monitor associated with the specified display, or a constant to indicate an exception—see the discussion below.

Discussion

This function uses I/O Kit to identify the monitor associated with the specified display.

There are three cases:

- If I/O Kit can identify the monitor, the vendor ID is returned.
- If I/O Kit cannot identify the monitor, `kDisplayVendorIDUnknown` is returned.
- If there is no monitor associated with the display, `0xFFFFFFFF` is returned.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayWaitForBeamPositionOutsideLines

Waits until the beam position moves outside a region in a display screen. This function is not designed for VBL drawing synchronization.

```
CGDisplayErr CGDisplayWaitForBeamPositionOutsideLines (
   CGDirectDisplayID display,
   CGBeamPosition upperScanLine,
   CGBeamPosition lowerScanLine
);
```

Parameters

display
The display to access.

upperScanLine
The upper scan line number.

lowerScanLine
The lower scan line number.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
This function waits until the beam position is outside the range specified by the arguments `upperScanLine` and `lowerScanLine`. If the value of `upperScanLine` is greater than the value of `lowerScanLine`, or if `upperScanLine` and `lowerScanLine` encompass the entire display height, this function returns an error.

Some displays may not use conventional video vertical and horizontal sweep in painting. These displays report a `kCGDisplayRefreshRate` of 0 in the dictionary returned by `CGDisplayCurrentMode` (page 32). Also, some display device drivers may not implement support for this mechanism. On such displays, this function returns at once.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGGetActiveDisplayList

Provides a list of displays that are active (or drawable).

```
CGDisplayErr CGGetActiveDisplayList (
   CGDisplayCount maxDisplays,
   CGDirectDisplayID *activeDspys,
   CGDisplayCount *dspyCnt
);
```

Parameters

maxDisplays

The size of the `activeDspys` array. This value determines the maximum number of displays that can be returned.

activeDspys

A pointer to storage provided by the caller for an array of display IDs. On return, the array contains a list of active displays. If you pass `NULL`, on return the display count contains the total number of active displays.

dspyCnt

A pointer to a display count variable provided by the caller. On return, the display count contains the actual number of displays returned in the `activeDspys` array. This value is at most `maxDisplays`.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion

The first entry in the list of active displays is the main display. In case of mirroring, the first entry is the largest drawable display or, if all are the same size, the display with the greatest pixel depth.

Note that when hardware mirroring is being used between displays, only the primary display is active and appears in the list. When software mirroring is being used, all the mirrored displays are active and appear in the list. For more information about mirroring, see `CGConfigureDisplayMirrorOfDisplay` (page 19).

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

QTCarbonShell

Declared In

`CGDirectDisplay.h`

CGGetDisplaysWithOpenGLDisplayMask

Provides a list of displays that corresponds to the bits set in an OpenGL display mask.

```
CGDisplayErr CGGetDisplaysWithOpenGLDisplayMask (
   CGOpenGLDisplayMask mask,
   CGDisplayCount maxDisplays,
   CGDirectDisplayID *dspys,
   CGDisplayCount *dspyCnt
);
```

Parameters

mask

An OpenGL display mask that identifies one or more displays.

maxDisplays

The size of the `dspys` array. This value determines the maximum number of displays that can be returned.

dspys

A pointer to storage provided by the caller for an array of display IDs. On return, the array contains a list of displays that corresponds to the bits set in the mask. If you pass `NULL`, on return the display count contains the total number of displays specified in the mask.

dspyCnt

A pointer to a display count variable provided by the caller. On return, the display count contains the actual number of displays returned in the `dspys` array. This value is at most `maxDisplays`.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGGetDisplaysWithPoint

Provides a list of online displays with bounds that include the specified point.

```
CGDisplayErr CGGetDisplaysWithPoint (
   CGPoint point,
   CGDisplayCount maxDisplays,
   CGDirectDisplayID *dspys,
   CGDisplayCount *dspyCnt
);
```

Parameters

point

The coordinates of a point in global display space. The origin is the upper left corner of the main display.

maxDisplays

The size of the `dspys` array. This value determines the maximum number of displays that can be returned.

dspys

A pointer to storage provided by the caller for an array of display IDs. On return, the array contains a list of displays with bounds that include the point. If you pass `NULL`, on return the display count contains the total number of displays with bounds that include the point.

dspyCnt

A pointer to a display count variable provided by the caller. On return, the display count contains the actual number of displays returned in the `dspys` array. This value is at most `maxDisplays`.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGGetDisplaysWithRect

Gets a list of online displays with bounds that intersect the specified rectangle.

```
CGDisplayErr CGGetDisplaysWithRect (
   CGRect rect,
   CGDisplayCount maxDisplays,
   CGDirectDisplayID *dspys,
   CGDisplayCount *dspyCnt
);
```

Parameters

rect

The location and size of a rectangle in global display space. The origin is the upper left corner of the main display.

maxDisplays

The size of the *dspys* array. This value determines the maximum number of displays that can be returned in the *dspys* parameter. Generally, you should specify a number greater than 0 for this parameter. If you specify 0, the value returned in *dspyCnt* is undefined and this function sets the *dspys* parameter to NULL.

dspys

A pointer to storage provided by the caller for an array of display IDs. On return, the array contains a list of displays whose bounds intersect the specified rectangle.

dspyCnt

A pointer to a display count variable provided by the caller. On return, this variable contains the number of displays that were returned in the *dspys* parameter. You must provide a non-NULL value for this parameter.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGGetDisplayTransferByFormula

Gets the coefficients of the gamma transfer formula for a display.

```
CGDisplayErr CGGetDisplayTransferByFormula (
   CGDirectDisplayID display,
   CGGammaValue *redMin,
   CGGammaValue *redMax,
   CGGammaValue *redGamma,
   CGGammaValue *greenMin,
   CGGammaValue *greenMax,
   CGGammaValue *greenGamma,
   CGGammaValue *blueMin,
   CGGammaValue *blueMax,
   CGGammaValue *blueGamma
);
```

Parameters

display

The display to access.

redMin

The minimum value of the red channel in the gamma table. The value is a number in the interval [0, redMax).

redMax

The maximum value of the red channel in the gamma table. The value is a number in the interval (redMin, 1].

redGamma

A positive value used to compute the red channel in the gamma table.

greenMin

The minimum value of the green channel in the gamma table. The value is a number in the interval [0, greenMax).

greenMax

The maximum value of the green channel in the gamma table. The value is a number in the interval (greenMin, 1].

greenGamma

A positive value used to compute the green channel in the gamma table.

blueMin

The minimum value of the blue channel in the gamma table. The value is a number in the interval [0, blueMax).

blueMax

The maximum value of the blue channel in the gamma table. The value is a number in the interval (blueMin, 1].

blueGamma

A positive value used to compute the blue channel in the gamma table.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion

For information about the gamma transfer formula, see the description of the function `CGSetDisplayTransferByFormula` (page 71).

Availability

Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGGetDisplayTransferByTable

Gets the values in the RGB gamma tables for a display.

```
CGDisplayErr CGGetDisplayTransferByTable (
   CGDirectDisplayID display,
   CGTableCount capacity,
   CGGammaValue *redTable,
   CGGammaValue *greenTable,
   CGGammaValue *blueTable,
   CGTableCount *sampleCount
);
```

Parameters

display
The display to access.

capacity
The number of entries each table can hold.

redTable
A pointer to an array of type `CGGammaValue` with size `capacity`. On return, the array contains the values of the red channel in the display's gamma table.

greenTable
A pointer to an array of type `CGGammaValue` with size `capacity`. On return, the array contains the values of the green channel in the display's gamma table.

blueTable
A pointer to an array of type `CGGammaValue` with size `capacity`. On return, the array contains the values of the blue channel in the display's gamma table.

sampleCount
The number of samples actually copied into each array.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGGetLastMouseDelta

Reports the change in mouse position since the last mouse movement event received by the application.

```
void CGGetLastMouseDelta (
   CGMouseDelta *deltaX,
   CGMouseDelta *deltaY
);
```

Parameters

deltaX

A pointer to a `CGMouseDelta` variable. On return, this variable contains the horizontal change in the mouse position since the last mouse movement event.

deltaY

A pointer to a `CGMouseDelta` variable. On return, this variable contains the vertical change in the mouse position since the last mouse movement event.

Discussion

This function is not recommended for general use. Instead, you should use the mouse tracking functions in the Carbon Event Manager.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGGetOnlineDisplayList

Provides a list of displays that are online (active, mirrored, or sleeping).

```
CGDisplayErr CGGetOnlineDisplayList (
   CGDisplayCount maxDisplays,
   CGDirectDisplayID *onlineDspys,
   CGDisplayCount *dspyCnt
);
```

Parameters

maxDisplays

The size of the `onlineDspys` array. This value determines the maximum number of display IDs that can be returned.

onlineDspys

A pointer to storage provided by the caller for an array of display IDs. On return, the array contains a list of the online displays. If you pass `NULL`, on return the display count contains the total number of online displays.

dspyCnt

A pointer to a display count variable provided by the caller. On return, the display count contains the actual number of displays returned in the `onlineDspys` array. This value is at most `maxDisplays`.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion

If the frame buffer hardware is connected, a display is considered connected or online.

When hardware mirroring is used, a display can be online but not active or drawable. Programs which manipulate display settings such as the palette or gamma tables need access to all displays, including hardware mirrors which are not drawable.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDirectDisplay.h`

CGMainDisplayID

Returns the display ID of the main display.

```
CGDirectDisplayID CGMainDisplayID (
   void
);
```

Return Value
The display ID assigned to the main display.

Discussion
The main display is the display with its screen location at (0,0) in global coordinates. In a system without display mirroring, the display with the menu bar is typically the main display.

If mirroring is enabled and the menu bar appears on more than one display, this function provides a reliable way to find the main display.

In case of hardware mirroring, the drawable display becomes the main display. In case of software mirroring, the display with the highest resolution and deepest pixel depth typically becomes the main display.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
LiveVideoMixer2

Declared In
`CGDirectDisplay.h`

CGOpenGLDisplayMaskToDisplayID

Maps an OpenGL display mask to a display ID.

```
CGDirectDisplayID CGOpenGLDisplayMaskToDisplayID (
   CGOpenGLDisplayMask mask
);
```

Parameters

mask
The OpenGL display mask to be converted.

Return Value
The display ID assigned to the specified display mask, or `kCGNullDirectDisplay` if no display matches the mask.

Discussion
OpenGL sometimes identifies a display using a bitmask with one bit set. This function maps such a display mask to the corresponding display ID. If you pass in a mask with multiple bits set, this function returns a display ID matching one of these bits.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDirectDisplay.h`

CGPaletteCreateCopy

Returns a copy of a specified display palette.

```
CGDirectPaletteRef CGPaletteCreateCopy (
   CGDirectPaletteRef palette
);
```

Parameters

palette
The display palette to copy.

Return Value
A new display palette object. When you no longer need the palette, you should release it using the function `CGPaletteRelease` (page 66).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectPalette.h`

CGPaletteCreateDefaultColorPalette

Returns a new display palette representing the default 8-bit color palette.

```
CGDirectPaletteRef CGPaletteCreateDefaultColorPalette (
   void
);
```

Return Value
A new display palette object. When you no longer need the palette, you should release it using the function `CGPaletteRelease` (page 66).

Discussion
Palettes are used with 256 color display modes. The default palette is the old default 8-bit Mac OS palette, with white at index 0 and black at index 255.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectPalette.h`

CGPaletteCreateFromPaletteBlendedWithColor

Returns a new tinted display palette. The new palette is derived from an existing palette blended with a solid color, at a specified level of intensity.

```
CGDirectPaletteRef CGPaletteCreateFromPaletteBlendedWithColor (
   CGDirectPaletteRef palette,
   CGPaletteBlendFraction fraction,
   CGDeviceColor color
);
```

Parameters

palette

The palette to blend.

fraction

A value between 0 and 1 that represents the blend intensity. See `CGPaletteBlendFraction` (page 89).

color

The blend color. See `CGDeviceColor` (page 83).

Return Value

A new display palette object. When you no longer need the palette, you should release it using the function `CGPaletteRelease` (page 66).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectPalette.h`

CGPaletteCreateWithByteSamples

Returns a new display palette using 8-bit sample data.

```
CGDirectPaletteRef CGPaletteCreateWithByteSamples (
   CGDeviceByteColor *sampleTable,
   CGTableCount sampleCount
);
```

Parameters

sampleTable

A color table with integer values that represent the intensity of the red,green, and blue components in each table entry. Each value ranges from 0 (no color) to 255 (full intensity). See `CGDeviceByteColor` (page 82).

sampleCount

The number of entries in the specified color table.

Return Value

A new display palette object. When you no longer need the palette, you should release it using the function `CGPaletteRelease` (page 66).

Availability

Available in Mac OS X v10.0 and later.

Declared In
CGDirectPalette.h

CGPaletteCreateWithCapacity

Returns a new display palette with a specified capacity. The new palette is initialized from the default color palette.

```
CGDirectPaletteRef CGPaletteCreateWithCapacity (
   CGTableCount capacity
);
```

Parameters

capacity
The number of entries in the new palette.

Return Value
A new display palette object. When you no longer need the palette, you should release it using the function CGPaletteRelease (page 66).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectPalette.h

CGPaletteCreateWithDisplay

Returns a copy of the current palette for a display.

```
CGDirectPaletteRef CGPaletteCreateWithDisplay (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
A new display palette object, or NULL if the current display mode does not support a palette. When you no longer need the palette, you should release it using the function CGPaletteRelease (page 66).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectPalette.h

CGPaletteCreateWithSamples

Returns a new display palette using RGB sample data.

```
CGDirectPaletteRef CGPaletteCreateWithSamples (
   CGDeviceColor *sampleTable,
   CGTableCount sampleCount
);
```

Parameters

sampleTable

A color table with floating point values that represent the intensity of the red,green, and blue components in each table entry. Each value ranges from 0 (no color) to 1 (full intensity). See `CGDeviceColor` (page 83).

sampleCount

The number of entries in the specified color table.

Return Value

A new display palette object. When you no longer need the palette, you should release it using the function `CGPaletteRelease` (page 66).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectPalette.h`

CGPaletteGetColorAtIndex

Returns the color value at the specified index.

```
CGDeviceColor CGPaletteGetColorAtIndex (
   CGDirectPaletteRef palette,
   CGTableCount index
);
```

Parameters

palette

The display palette to access.

index

The zero-based index of the desired palette entry.

Return Value

A color value. See `CGDeviceColor` (page 83).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectPalette.h`

CGPaletteGetIndexForColor

Returns the index of the display palette entry that most closely matches a specified color value.

```
CGTableCount CGPaletteGetIndexForColor (
   CGDirectPaletteRef palette,
   CGDeviceColor color
);
```

Parameters

palette

The display palette to access.

color

The color value to match. See `CGDeviceColor` (page 83).

Return Value

The index of the display palette entry that most closely matches the specified color value.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectPalette.h`

CGPaletteGetNumberOfSamples

Returns the number of colors in a display palette.

```
CGTableCount CGPaletteGetNumberOfSamples (
   CGDirectPaletteRef palette
);
```

Parameters

palette

The display palette to access.

Return Value

The number of colors in the specified display palette.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectPalette.h`

CGPaletteIsEqualToPalette

Returns a Boolean value indicating whether two display palettes are equal.

```
Boolean CGPaletteIsEqualToPalette (
   CGDirectPaletteRef palette1,
   CGDirectPaletteRef palette2
);
```

Parameters

palette1

The first display palette to compare.

palette2
The second display palette to compare.

Return Value
If `true`, the two specified display palettes are equal; otherwise, `false`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectPalette.h`

CGPaletteRelease

Decrements the retain count of a display palette.

```
void CGPaletteRelease (
   CGDirectPaletteRef palette
);
```

Parameters

palette
The display palette to release.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectPalette.h`

CGPaletteSetColorAtIndex

Updates the color value at the specified index in a display palette.

```
void CGPaletteSetColorAtIndex (
   CGDirectPaletteRef palette,
   CGDeviceColor color,
   CGTableCount index
);
```

Parameters

palette
The display palette to access.

color
The new color value.

index
The index of the palette entry to update.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectPalette.h`

CGRegisterScreenRefreshCallback

Registers a callback function to be invoked when local displays are refreshed or modified.

```
CGError CGRegisterScreenRefreshCallback (
   CGScreenRefreshCallback function,
   void *userParameter
);
```

Parameters

function

A pointer to the callback function to be registered.

userParameter

A pointer to user-defined data, or `NULL`. The `userParameter` argument is passed back to the callback function each time it's invoked.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion

A callback function may be registered multiple times with different user-defined data pointers, resulting in multiple registration entries. For each registration, when notification is no longer needed you should call the function `CGUnregisterScreenRefreshCallback` (page 74) to remove the registration.

The callback function you register is invoked only if your application has an active event loop. The callback is invoked in the same thread of execution that is processing events within your application.

Special Considerations

In Mac OS X v10.4 and earlier, the result code returned by this function is a random value and should be ignored. In Mac OS X v10.5 and later, the result code is valid.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGRemoteOperation.h`

CGReleaseAllDisplays

Releases all captured displays.

```
CGDisplayErr CGReleaseAllDisplays (
   void
);
```

Return Value

A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion

This function releases all captured displays and restores the display modes to the user's preferences. It may be used in conjunction with any of the functions that capture displays, such as `CGCaptureAllDisplays` (page 16).

Availability

Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGReleaseDisplayFadeReservation

Releases a display fade reservation, and unfades the display if needed.

```
CGError CGReleaseDisplayFadeReservation (
   CGDisplayFadeReservationToken myToken
);
```

Parameters

myToken

The current fade reservation token to be released. On return, the reservation token is no longer valid and should be discarded.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
If you call this function while an asynchronous fade operation is running, there are two possible outcomes:

- If the ending blend value is `kCGDisplayBlendNormal`, the fade operation is allowed to run to completion.
- If the ending blend value is not `kCGDisplayBlendNormal`, the fade operation is terminated immediately and the display is returned to normal.

In both cases, the reservation is actually released when the fade operation completes.

Availability
Available in Mac OS X v10.2 and later.

Declared In
CGDisplayFade.h

CGReleaseScreenRefreshRects

Deallocates a list of rectangles that represent changed areas on local displays.

```
void CGReleaseScreenRefreshRects (
   CGRect *rectArray
);
```

Parameters

rectArray

A list of rectangles obtained by calling `CGWaitForScreenRefreshRects` (page 75) or `CGWaitForScreenUpdateRects` (page 76).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGRemoteOperation.h

CGRestorePermanentDisplayConfiguration

Restores the permanent display configuration settings for the current user.

```
void CGRestorePermanentDisplayConfiguration (
   void
);
```

Discussion
This function provides a convenient way to restore the permanent display configuration.

Applications that temporarily change the display configuration—such as applications and games that switch to full-screen display mode—can use this function to undo the changes.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGScreenRegisterMoveCallback

Registers a callback function to be invoked when an area of the display is moved.

```
CGError CGScreenRegisterMoveCallback (
   CGScreenUpdateMoveCallback function,
   void *userParameter
);
```

Parameters

function
: A pointer to the callback function to be registered.

userParameter
: A pointer to user-defined data, or `NULL`. The `userParameter` argument is passed back to the callback function each time it's invoked.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
A callback function may be registered multiple times with different user-defined data pointers, resulting in multiple registration entries. For each registration, when notification is no longer needed you should remove the registration by calling the function `CGScreenUnregisterMoveCallback` (page 70).

The callback function you register is invoked only if your application has an active event loop. The callback is invoked in the same thread of execution that is processing events within your application.

Special Considerations

This function is implemented in Mac OS X version 10.4.3 and later.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGRemoteOperation.h`

CGScreenUnregisterMoveCallback

Removes a previously registered callback function invoked when an area of the display is moved.

```
void CGScreenUnregisterMoveCallback (
   CGScreenUpdateMoveCallback function,
   void *userParameter
);
```

Parameters

function

A pointer to the callback function to be unregistered.

userParameter

A pointer to user-defined data, or `NULL`. You should pass the same value you used when you registered the callback function.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion

When you call this function, the two arguments must match the registered entry to be removed.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGRemoteOperation.h`

CGSessionCopyCurrentDictionary

Returns information about the caller's window server session.

```
CFDictionaryRef CGSessionCopyCurrentDictionary (
   void
);
```

Return Value

A window server session dictionary, or `NULL` if the caller is not running within a Quartz GUI session or the window server is disabled. You should release the dictionary when you are finished using it. For information about the key-value pairs in this dictionary, see "Window Server Session Properties" (page 101).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGSession.h`

CGSetDisplayTransferByByteTable

Sets the byte values in the 8-bit RGB gamma tables for a display.

```
CGDisplayErr CGSetDisplayTransferByByteTable (
   CGDirectDisplayID display,
   CGTableCount tableSize,
   const CGByteValue *redTable,
   const CGByteValue *greenTable,
   const CGByteValue *blueTable
);
```

Parameters

display

The display to access.

tableSize

The number of entries in each table.

redTable

An array of size `tableSize` containing the byte values of the red channel in the display's gamma table.

greenTable

An array of size `tableSize` containing the byte values of the green channel in the display's gamma table.

blueTable

An array of size `tableSize` containing the byte values of the blue channel in the display's gamma table.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion

The same table may be passed in for the red, green, and blue channels. The tables are interpolated as needed to generate the number of samples required by the graphics hardware.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGSetDisplayTransferByFormula

Sets the gamma function for a display, by specifying the coefficients of the gamma transfer formula.

```
CGDisplayErr CGSetDisplayTransferByFormula (
   CGDirectDisplayID display,
   CGGammaValue redMin,
   CGGammaValue redMax,
   CGGammaValue redGamma,
   CGGammaValue greenMin,
   CGGammaValue greenMax,
   CGGammaValue greenGamma,
   CGGammaValue blueMin,
   CGGammaValue blueMax,
   CGGammaValue blueGamma
);
```

Parameters

display

The display to access.

redMin

The minimum value of the red channel in the gamma table. The value should be a number in the interval [0, redMax).

redMax

The maximum value of the red channel in the gamma table. The value should be a number in the interval (redMin, 1].

redGamma

A positive value used to compute the red channel in the gamma table.

greenMin

The minimum value of the green channel in the gamma table. The value should be a number in the interval [0, greenMax).

greenMax

The maximum value of the green channel in the gamma table. The value should be a number in the interval (greenMin, 1].

greenGamma

A positive value used to compute the green channel in the gamma table.

blueMin

The minimum value of the blue channel in the gamma table. The value should be a number in the interval [0, blueMax).

blueMax

The maximum value of the blue channel in the gamma table. The value should be a number in the interval (blueMin, 1].

blueGamma

A positive value used to compute the blue channel in the gamma table.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion

This function uses the specified parameter values to compute a gamma correction table for the specified display. The values in the table are computed by sampling the following gamma transfer formula for a range of indices from 0 to 1:

```
value = Min + ((Max - Min) * pow(index, Gamma))
```

The resulting values are converted to a machine-specific format and loaded into display hardware.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGSetDisplayTransferByTable

Sets the color gamma function for a display, by specifying the values in the RGB gamma tables.

```
CGDisplayErr CGSetDisplayTransferByTable (
   CGDirectDisplayID display,
   CGTableCount tableSize,
   const CGGammaValue *redTable,
   const CGGammaValue *greenTable,
   const CGGammaValue *blueTable
);
```

Parameters

display
The display to access.

tableSize
The number of entries in each table.

redTable
An array of size `tableSize` containing the values of the red channel in the display's gamma table. The values should be in the range 0.0 to 1.0.

greenTable
An array of size `tableSize` containing the values of the green channel in the display's gamma table. The values should be in the range 0.0 to 1.0.

blueTable
An array of size `tableSize` containing the values of the blue channel in the display's gamma table. The values should be in the range 0.0 to 1.0.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
The same table may be passed in for the red, green, and blue channels. The tables are interpolated as needed to generate the number of samples required by the graphics hardware.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGShieldingWindowID

Returns the window ID of the shield window for a captured display.

```
uint32_t CGShieldingWindowID (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

The window ID of the shield window for the specified display, or `NULL` if the display is not shielded.

Discussion

To prevent updates by direct-to-screen programs (such as Classic), Quartz draws a shield window that fills the entire screen of a captured display.

This function is not recommended for use in applications. Note that the graphics context associated with this window is not a full-featured drawing context. To get a full-featured drawing context for a captured display, you should use the function `CGDisplayGetDrawingContext` (page 35).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGShieldingWindowLevel

Returns the window level of the shield window for a captured display.

```
int32_t CGShieldingWindowLevel (
   void
);
```

Return Value

The window level of the shield window for a captured display.

Discussion

This function returns a value that is sometimes used to position a window over the shield window for a captured display. Attempting to position a window over a captured display may be unsuccessful—or may present undesirable results such as illegible or invisible content—because of interactions between full-screen graphics (such as OpenGL full-screen drawing contexts) and the graphics hardware. Because of these limitations, and because the implementation of display capture may change in the future, this technique is not recommended.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGUnregisterScreenRefreshCallback

Removes a previously registered callback function invoked when local displays are refreshed or modified.

```
void CGUnregisterScreenRefreshCallback (
   CGScreenRefreshCallback function,
   void *userParameter
);
```

Parameters

function

A pointer to the callback function to be unregistered.

userParameter

A pointer to user-defined data, or `NULL`. You should pass the same value you used when you registered the callback function.

Discussion

When you call this function, the two arguments must match the registered entry to be removed.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGRemoteOperation.h`

CGWaitForScreenRefreshRects

Waits for screen refresh operations.

```
CGError CGWaitForScreenRefreshRects (
   CGRect **pRectArray,
   CGRectCount *pCount
);
```

Parameters

pRectArray

A pointer to a `CGRect*` variable. On return, the variable contains an array of rectangles that bound the refreshed areas, specified in global coordinates. When you no longer need the array, you should deallocate it by calling `CGReleaseScreenRefreshRects` (page 68).

pCount

A pointer to a `CGRectCount` variable. On return, the variable contains the number of entries in the returned array of rectangles.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion

In some applications it may be preferable to wait for screen refresh data synchronously, using this function. You should call this function in a thread other than the main event-processing thread.

As an alternative, Quartz also supports asynchronous notification—see `CGRegisterScreenRefreshCallback` (page 67). If refresh callback functions are registered, this function should not be used.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGRemoteOperation.h`

CGWaitForScreenUpdateRects

Waits for screen update operations.

```
CGError CGWaitForScreenUpdateRects (
   CGScreenUpdateOperation requestedOperations,
   CGScreenUpdateOperation *currentOperation,
   CGRect **pRectArray,
   size_t *pCount,
   CGScreenUpdateMoveDelta *pDelta
);
```

Parameters

requestedOperations

The desired types of screen update operations. There are several possible choices:

- Specify `kCGScreenUpdateOperationRefresh` if you want all move operations to be returned as refresh operations.
- Specify `(kCGScreenUpdateOperationRefresh | kCGScreenUpdateOperationMove)` if you want to distinguish between move and refresh operations.
- Add `kCGScreenUpdateOperationReducedDirtyRectangleCount` to the screen operations if you want to minimize the number of rectangles returned to represent changed areas of the display.

currentOperation

A pointer to a `CGScreenUpdateOperation` variable. On return, the variable indicates the type of update operation (refresh or move).

pRectArray

A pointer to a `CGRect*` variable. On return, the variable contains an array of rectangles that bound the updated areas, specified in global coordinates. When you no longer need the array, you should deallocate it by calling `CGReleaseScreenRefreshRects` (page 68).

pCount

A pointer to a `size_t` variable. On return, the variable contains the number of entries in the returned array of rectangles.

pDelta

A pointer to a `CGScreenUpdateMoveDelta` variable. On return, if the value of the `currentOperation` parameter is `kCGScreenUpdateOperationMove` the variable contains the distance moved.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion

In some applications it may be preferable to wait for screen update data synchronously, using this function. You should call this function in a thread other than the main event-processing thread.

As an alternative, Quartz also supports asynchronous notification—see `CGRegisterScreenRefreshCallback` (page 67) and `CGScreenRegisterMoveCallback` (page 69). If refresh or move callback functions are registered, this function should not be used.

Special Considerations

This function is implemented in Mac OS X version 10.4.3 and later.

Availability

Available in Mac OS X v10.3 and later.

Declared In
`CGRemoteOperation.h`

CGWarpMouseCursorPosition

Moves the mouse cursor without generating events.

```
CGError CGWarpMouseCursorPosition (
   CGPoint newCursorPosition
);
```

Parameters

`newCursorPosition`
The new mouse cursor position in global display coordinates.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 102).

Discussion
You can use this function to 'warp' or alter the cursor position without generating or posting an event. For example, this function is often used to move the cursor position back to the center of the screen by games that do not want the cursor pinned by display edges.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGRemoteOperation.h`

CGWindowLevelForKey

Returns the window level that corresponds to one of the standard window types.

```
CGWindowLevel CGWindowLevelForKey (
   CGWindowLevelKey key
);
```

Parameters

`key`
A window level key constant that represents one of the standard window types. See "Window Level Keys" (page 98).

Return Value
The window level that corresponds to the specified key.

Discussion
This function is not recommended for use in applications. (This function is provided for application frameworks that create and manage windows, such as Carbon and Cocoa.)

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGWindowLevel.h`

CGWindowServerCFMachPort

Returns a Core Foundation mach port (CFMachPort) that corresponds to the Mac OS X window server.

```
CFMachPortRef CGWindowServerCFMachPort (
   void
);
```

Return Value
A Core Foundation mach port, or `NULL` if the window server is not running. When you no longer need the port, you should release it using the function `CFRelease`.

Discussion
You can use this function to detect if the window server process exits or is not running. If this function returns `NULL`, the window server is not running. This code example shows how to register a callback function to detect when the window server exits:

```
static void handleWindowServerDeath( CFMachPortRef port, void *info  )
{
    printf( "Window Server port death detected!\n" );
    CFRelease(port);
    exit(1);
}

static void watchForWindowServerDeath()
{
    CFMachPortRef port = CGWindowServerCFMachPort();
    CFMachPortSetInvalidationCallBack(port, handleWindowServerDeath);
}
```

Note that this callback will not work unless your program has an active run loop.

Availability
Available in Mac OS X v10.1 and later.

Declared In
`CGRemoteOperation.h`

Callbacks

CGDisplayReconfigurationCallBack

A client-supplied callback function that's invoked whenever the configuration of a local display is changed.

```
typedef void (*CGDisplayReconfigurationCallBack) (
   CGDirectDisplayID display,
   CGDisplayChangeSummaryFlags flags,
   void *userInfo
);
```

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

```
void MyDisplayReconfigurationCallBack (
   CGDirectDisplayID display,
```

```
   CGDisplayChangeSummaryFlags flags,
   void *userInfo
);
```

Parameters

display

The display being reconfigured.

flags

Flags that indicate which display configuration parameters are changing.

userInfo

The `userInfo` argument passed to the function `CGDisplayRegisterReconfigurationCallback` (page 45) when the callback function is registered.

Discussion

To register a display reconfiguration callback function, you call the function `CGDisplayRegisterReconfigurationCallback` (page 45). Quartz invokes your callback function when:

- Your application calls a function to reconfigure a local display.
- Your application is listening for events in the event-processing thread, and another application calls a function to reconfigure a local display.
- The user changes the display hardware configuration—for example, by disconnecting a display or changing a system preferences setting.

Before display reconfiguration, Quartz invokes your callback function once for each online display to indicate a pending configuration change. The `flags` argument is always set to `kCGDisplayBeginConfigurationFlag`. Other than the display ID, this callback does not carry other per-display information, as details of how a reconfiguration affects a particular device rely on device-specific behaviors which may not be exposed by a device driver.

After display reconfiguration, Quartz invokes your callback function once for each added, removed, and online display. At this time, all display state reported by Core Graphics, QuickDraw, and the Carbon Display Manager will be up to date. This callback runs after the Carbon Display Manager notification callbacks. The `flags` argument indicates how the display configuration has changed. Note that in the case of removed displays, calls into Quartz with the removed display ID will fail.

The following code example illustrates how to test for specific conditions:

```
void MyDisplayReconfigurationCallBack (
   CGDirectDisplayID display,
   CGDisplayChangeSummaryFlags flags,
   void *userInfo)
{
    if (flags & kCGDisplayAddFlag) {
        // display has been added
    }
    else if (flags & kCGDisplayRemoveFlag) {
        // display has been removed
    }
}
```

Your callback function should avoid attempting to change display configurations, and should not raise exceptions or perform a non-local return such as calling `longjmp`. When you are finished using a callback registration, you should call the function `CGDisplayRemoveReconfigurationCallback` (page 46) to remove it.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGDisplayConfiguration.h`

CGScreenRefreshCallback

A client-supplied callback function that's invoked when an area of the display is modified or refreshed.

```
typedef void (*CGScreenRefreshCallback) (
   CGRectCount count,
   const CGRect * rectArray,
   void * userParameter
);
```

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

```
void MyScreenRefreshCallback (
   CGRectCount count,
   const CGRect * rectArray,
   void * userParameter
);
```

Parameters

count
The number of rectangles in the `rectArray` parameter.

rectArray
A list of the rectangles in the refreshed areas, specified in global coordinates. You should not modify or deallocate memory pointed to by `rectArray`.

userParameter
The user data you specify when you register this callback.

Discussion
To register a screen refresh callback function, you call the function `CGRegisterScreenRefreshCallback` (page 67). Quartz invokes your callback function when operations such as drawing, window movement, scrolling, or display reconfiguration occur on local displays. When you are finished using a callback registration, you should call the function `CGUnregisterScreenRefreshCallback` (page 74) to remove it.

Note that a single rectangle may occupy multiple displays, either by overlapping the displays or by residing on coincident displays when mirroring is active. You can use the function `CGGetDisplaysWithRect` (page 56) to determine the displays a rectangle occupies.

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGRemoteOperation.h

CGScreenUpdateMoveCallback

A client-supplied callback function that's invoked when an area of the display is moved.

```
typedef void (*CGScreenUpdateMoveCallback) (
   CGScreenUpdateMoveDelta delta,
   CGRectCount count,
   const CGRect * rectArray,
   void * userParameter
);
```

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

```
void MyScreenUpdateMoveCallback (
   CGScreenUpdateMoveDelta delta,
   CGRectCount count,
   const CGRect * rectArray,
   void * userParameter
);
```

Parameters

delta

The distance the display area has moved.

count

The number of rectangles in the `rectArray` parameter.

rectArray

A list of the rectangles in the moved areas, specified in global coordinates. The rectangles describe the area prior to the move operation. You should not modify or deallocate memory pointed to by `rectArray`.

userParameter

The user data you specify when you register this callback.

Discussion

To register a screen move callback function, you call the function `CGScreenRegisterMoveCallback` (page 69). Quartz invokes your callback function when operations such as window movement or scrolling occur on local displays. When you are finished using a callback registration, you should call the function `CGScreenUnregisterMoveCallback` (page 70) to remove it.

Note that a single rectangle may occupy multiple displays, either by overlapping the displays or by residing on coincident displays when mirroring is active. You can use the function `CGGetDisplaysWithRect` (page 56) to determine the displays a rectangle occupies.

Availability

Available in Mac OS X v10.3 and later.

Declared In
CGRemoteOperation.h

Data Types

CGBeamPosition

Represents a horizontal scan line on a monitor that uses a scanning electron beam to refresh the screen.

```
typedef uint32_t CGBeamPosition;
```

Discussion
CRT and analog-driven displays use a horizontal scanning beam to refresh the screen. The beam position is a number assigned to a horizontal scan line on the screen. Scan lines are numbered `0` to `n-1` from top of screen, where `n` represents the total number of scan lines.

The concept of beam position does not apply to flat-panel LCD displays. While all displays have some concept of scan lines with respect to the frame buffer, LCD displays may not use linear scanning to refresh the screen.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGByteValue

Represents a unit of information in a byte-addressable array or data structure.

```
typedef uint8_t CGByteValue;
```

Discussion
Quartz uses `CGByteValue` to represent integer-based color values in a display palette or a gamma table. For example, see `CGDeviceByteColor` (page 82).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDeviceByteColor

Represents a color in a Quartz display palette, using 8-bit integer components.

```
struct CGDeviceByteColor {
    CGByteValue red;
    CGByteValue green;
    CGByteValue blue;
};
typedef struct CGDeviceByteColor CGDeviceByteColor;
```

Fields
`red`
The red component of a palette entry.

`green`
: The green component of a palette entry.

`blue`
: The blue component of a palette entry.

Discussion
This data structure consists of three integer values that represent the intensity of the red,green, and blue components in a display palette entry. Each component ranges from 0 (no color) to 255 (full intensity).

Quartz provides `CGDeviceByteColor` to allow you to create a display palette using integer-based sample data. Once loaded, you can retrieve color data from the palette only as entries of type `CGDeviceColor` (page 83).

For more information about display palettes, see `CGDirectPaletteRef` (page 84).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectPalette.h`

CGDeviceColor

Represents a color in a Quartz display palette.

```
struct CGDeviceColor {
    float red;
    float green;
    float blue;
};
typedef struct CGDeviceColor CGDeviceColor;
```

Fields
`red`
: The red component of a palette entry.

`green`
: The green component of a palette entry.

`blue`
: The blue component of a palette entry.

Discussion
This data structure consists of three floating point values that represent the intensity of the red,green, and blue components in a display palette entry. Each component ranges from 0 (no color) to 1 (full intensity). Values outside this range are clamped to 0 or 1 when the palette is created.

Quartz uses `CGDeviceColor` as the canonical form for a color entry in a display palette. Palette entries can be created and retrieved in this form.

For more information about display palettes, see `CGDirectPaletteRef` (page 84).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectPalette.h`

CGDirectDisplayID

Represents a unique identifier for an attached display.

```
typedef uint32_t CGDirectDisplayID;
```

Discussion
In Quartz, the term *display* refers to a graphics hardware system consisting of a framebuffer, a color correction (gamma) table or color palette, and possibly an attached monitor. If no monitor is attached, a display is characterized as offline.

When a monitor is attached, Quartz assigns a unique display identifier (ID). A display ID can persist across processes and system reboot, and typically remains constant as long as certain display parameters do not change.

When assigning a display ID, Quartz considers the following parameters:

- vendor
- model
- serial number
- position in the I/O Kit registry

For information about how to obtain a display ID, see "Finding Displays" (page 8).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDirectPaletteRef

Defines a reference to a Quartz 8-bit display palette.

```
typedef struct _CGDirectPaletteRef * CGDirectPaletteRef;
```

Discussion
A display palette is a bounded set of color values available for display. Some display operating modes have a maximum color depth of 8 bits (256 colors). The CGDirectPalette API is designed for application and game developers that want to create and use display palettes for these older displays.

Quartz uses reference counting to manage display palettes. See "Working With Color Palettes" (page 12) for more information.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayBlendFraction

Represents the percentage of blend color used in a fade operation.

```
typedef float CGDisplayBlendFraction;
```

Discussion
The blend fraction ranges from 0 (no color) to 1 (full intensity). If you specify 0, the blend color is not applied. If you specify 1, the user sees only the blend color on the screen.

In a fade operation, Quartz blends a color specified by the application with the current contents of the frame buffer. The blend color can be applied both at the beginning and the end of a fade operation.

Color blending during a fade operation is analogous to alpha blending in Quartz 2D, and the visual appearance is similar. However, the implementation is quite different. In a fade operation, the blend color is applied at the very end of the graphics pipeline, as the frame buffer is transferred to video output.

For example, the Universal Access preference panel in Mac OS X allows you to select a flashing screen effect (sometimes called a visual bell) to accompany the system alert sound. When you select this option, the system uses a Quartz fade operation to produce the flash. The blend color is applied using a blend fraction of 0.5 or 50%.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGDisplayConfigRef

Defines a reference to a display configuration transaction.

```
typedef struct _CGDisplayConfigRef * CGDisplayConfigRef;
```

Discussion
This data type makes it possible to

- create a new display configuration transaction using the function `CGBeginDisplayConfiguration` (page 15)
- record a set of configuration changes, each bound to one or more displays
- apply the changes in a single transaction using the function `CGCompleteDisplayConfiguration` (page 17), or discard the changes using the function `CGCancelDisplayConfiguration` (page 16)

There are no restrictions on the order in which you accumulate configuration changes in a transaction.

Configuration changes sometimes conflict with each other. For example, a new origin might be rendered invalid by a subsequent configuration change.

If possible, Quartz uses a “best fit” strategy to resolve conflicts between configuration changes. For example, when you change the resolution of a single display in a two-display system, Quartz automatically re-tiles the displays to prevent separation or overlap of the adjoining edges.

Availability
Available in Mac OS X v10.2 and later.

Declared In
CGDisplayConfiguration.h

CGDisplayCoord

Represents a coordinate position in global display space.

```
typedef int32_t CGDisplayCoord;
```

Discussion
Quartz uses `CGDisplayCoord` to represent the x- and y-coordinates of points in the per-display coordinate system. The origin is defined as the upper-left corner of the screen.

This data type is also used in functions that need to find the address of a specific pixel and monitor. For example, the function `CGDisplayAddressForPosition` (page 23) finds the address in frame buffer memory that corresponds to a given position or point.

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGDisplayCount

Represents the number of displays in various lists.

```
typedef uint32_t CGDisplayCount;
```

Discussion
Quartz uses `CGDisplayCount` to represent a count of either the current or the maximum number of displays in a display list. For example, see the function `CGGetActiveDisplayList` (page 53).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGDisplayErr

Defines a uniform type for result codes returned by functions in Quartz Display Services.

```
typedef CGError CGDisplayErr;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGDisplayFadeInterval

Represents the duration in seconds of a fade operation or a fade hardware reservation.

```
typedef float CGDisplayFadeInterval;
```

Discussion
Quartz uses this data type to specify the duration of both fade-out and fade-in operations. Values may range from zero to `kCGMaxDisplayReservationInterval` seconds. A zero value means fade immediately—see `CGDisplayFade` (page 33).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGDisplayFadeReservationToken

Defines a token issued by Quartz when reserving one or more displays for a fade operation during a specified interval.

```
typedef uint32_t CGDisplayFadeReservationToken;
```

Discussion
Quartz lets you reserve the display hardware to perform a fade operation. Fade reservations are valid for up to 15 seconds. Only one token is needed for both fade-out and fade-in.

You should release a fade reservation immediately when you no longer need it. If the reservation expires, releasing it is safe but not necessary.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGDisplayReservationInterval

Represents the time interval for a fade reservation.

```
typedef float CGDisplayReservationInterval;
```

Discussion
A fade reservation interval is a period of time during which a specific display is reserved for a fade operation. Fade reservation intervals range from 1 to `kCGMaxDisplayReservationInterval` seconds.

For more information about fade reservations, see the function `CGAcquireDisplayFadeReservation` (page 14). Fade reservation tokens are discussed in `CGDisplayFadeReservationToken` (page 87).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGError

Defines a uniform type for result codes returned by functions in Quartz Services.

```
typedef int32_t CGError;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGError.h`

CGGammaValue

Represents information used to map a color generated in software to a color supported by the display hardware.

```
typedef float CGGammaValue;
```

Discussion
In Mac OS X, the Display panel in System Preferences is used to set the default gamma for a display. Quartz also allows an application to provide its own custom gamma information, using functions such as `CGSetDisplayTransferByTable` (page 73) and `CGSetDisplayTransferByFormula` (page 71).

These functions take `CGGammaValue` arguments that specify

- a set of gamma table entries ranging from 0 to 1
- the positive real coefficients in a gamma equation

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGMouseDelta

Represents a change in mouse position, in mouse units.

```
typedef int32_t CGMouseDelta;
```

Discussion
A mouse unit is a hardware-specific unit of measure, and generally has higher resolution than pixel units.

Note that the function `CGGetLastMouseDelta` (page 58) is no longer recommended—instead, you should use mouse tracking functions in the Carbon Event Manager.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGOpenGLDisplayMask

Defines a bitmask used in OpenGL to specify a set of attached displays.

```
typedef uint32_t CGOpenGLDisplayMask;
```

Discussion
In Mac OS X, OpenGL can provide information about the capabilities of the hardware renderers driving a specified set of displays. A 32-bit mask is used to specify the displays—each bit in the mask represents a single display.

To learn how to find the mask bit that corresponds to a given display, see the function `CGDisplayIDToOpenGLDisplayMask` (page 36).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGPaletteBlendFraction

Represents the intensity of a solid color used to tint a display palette.

```
typedef float CGPaletteBlendFraction;
```

Discussion
A palette blend-fraction value can range from 0 (no color) to 1 (full intensity). At full intensity, the palette is completely washed out by the color.

For more information, see the function `CGPaletteCreateFromPaletteBlendedWithColor` (page 62).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectPalette.h`

CGRectCount

Represents the size of an array of Quartz rectangles.

```
typedef uint32_t CGRectCount;
```

Discussion
For example, see the function `CGWaitForScreenRefreshRects` (page 75).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGRemoteOperation.h`

CGRefreshRate

Represents a display's refresh rate in frames per second.

```
typedef double CGRefreshRate;
```

Discussion
When requesting a new display mode, you can specify a desired refresh rate as a hint to Quartz. For example, see the function `CGDisplayBestModeForParametersAndRefreshRate` (page 26).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGScreenUpdateMoveDelta

Represents the distance a region on the screen moves in pixel units.

```
struct _CGScreenUpdateMoveDelta {
      int32_t dX, dY;
};
typedef struct _CGScreenUpdateMoveDelta CGScreenUpdateMoveDelta;
```

Discussion
Move operation notifications are restricted to changes that move a region by an integer number of pixels. The fields `dX` and `dY` describe the direction of movement:

- Positive values of `dX` indicate movement to the right.
- Negative values of `dX` indicate movement to the left.
- Positive values of `dY` indicate movement downward.
- Negative values of `dY` indicate movement upward.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGRemoteOperation.h`

CGTableCount

Defines a uniform type to represent the number of entries in a table.

```
typedef uint32_t CGTableCount;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGWindowLevel

Represents a level assigned to a window by an application framework.

```
typedef int32_t CGWindowLevel;
```

Discussion
In Mac OS X, application frameworks support the concept of multiple window levels (or layers). Window levels are assigned and managed by each individual framework.

Note that in an Aqua-compliant application, each document window exists in its own layer. As a result, windows created by different applications can be interleaved.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGWindowLevel.h`

Constants

Display Capture Options

Specify configuration parameters when capturing displays.

```
enum {
    kCGCaptureNoOptions = 0,
    kCGCaptureNoFill = (1 << 0)
};
typedef uint32_t CGCaptureOptions;
```

Constants

`kCGCaptureNoOptions`
Specifies that the system should use the default fill behavior, which is fill with black.
Available in Mac OS X v10.3 and later.
Declared in `CGDirectDisplay.h`.

`kCGCaptureNoFill`
Disables fill with black.
Available in Mac OS X v10.3 and later.
Declared in `CGDirectDisplay.h`.

Discussion
For information about how these constants are used, see the functions `CGDisplayCaptureWithOptions` (page 31) and `CGCaptureAllDisplaysWithOptions` (page 17).

Display Configuration Change Flags

Specify the configuration parameters passed to a display reconfiguration callback function.

```
enum {
    kCGDisplayBeginConfigurationFlag = (1 << 0),
    kCGDisplayMovedFlag              = (1 << 1),
    kCGDisplaySetMainFlag            = (1 << 2),
    kCGDisplaySetModeFlag            = (1 << 3),
    kCGDisplayAddFlag                = (1 << 4),
    kCGDisplayRemoveFlag             = (1 << 5),
    kCGDisplayEnabledFlag            = (1 << 8),
    kCGDisplayDisabledFlag           = (1 << 9),
    kCGDisplayMirrorFlag             = (1 << 10),
    kCGDisplayUnMirrorFlag           = (1 << 11),
    kCGDisplayDesktopShapeChangedFlag = (1 << 12)
};
typedef u_int32_t CGDisplayChangeSummaryFlags;
```

Constants

`kCGDisplayBeginConfigurationFlag`

The display configuration is about to change.

Available in Mac OS X v10.3 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGDisplayMovedFlag`

The location of the upper-left corner of the display in global display space has changed.

Available in Mac OS X v10.3 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGDisplaySetMainFlag`

The display is now the main display.

Available in Mac OS X v10.3 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGDisplaySetModeFlag`

The display mode has changed.

Available in Mac OS X v10.3 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGDisplayAddFlag`

The display has been added to the active display list.

Available in Mac OS X v10.3 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGDisplayRemoveFlag`

The display has been removed from the active display list.

Available in Mac OS X v10.3 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGDisplayEnabledFlag`

The display has been enabled.

Available in Mac OS X v10.3 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGDisplayDisabledFlag`
The display has been disabled.

Available in Mac OS X v10.3 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGDisplayMirrorFlag`
The display is now mirroring another display.

Available in Mac OS X v10.3 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGDisplayUnMirrorFlag`
The display is no longer mirroring another display.

Available in Mac OS X v10.3 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGDisplayDesktopShapeChangedFlag`
The shape of the desktop (the union of display areas) has changed.

Available in Mac OS X v10.5 and later.

Declared in `CGDisplayConfiguration.h`.

Discussion
For information about how these constants are used, see the callback `CGDisplayReconfigurationCallBack` (page 78).

Display Configuration Scopes

Specify the scope of the changes in a display configuration transaction.

```
enum {
    kCGConfigureForAppOnly = 0,
    kCGConfigureForSession = 1,
    kCGConfigurePermanently = 2
};
```

Constants
`kCGConfigureForAppOnly`
Specifies that changes persist for the lifetime of the current application. After the application terminates, the display configuration settings revert to the current login session.

Available in Mac OS X v10.2 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGConfigureForSession`
Specifies that changes persist for the lifetime of the current login session. After the current session terminates, the displays revert to the last saved permanent configuration.

Available in Mac OS X v10.2 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGConfigurePermanently`
: Specifies that changes persist in future login sessions by the same user. If the requested changes cannot be supported by the Aqua UI (resolution and pixel depth constraints apply), the settings for the current login session are used instead, and any changes have session scope.

  Available in Mac OS X v10.2 and later.

  Declared in `CGDisplayConfiguration.h`.

Discussion
For information about how these constants are used, see the function `CGCompleteDisplayConfiguration` (page 17).

Display Fade Blend Fractions

Specify the lower and upper bounds for blend color fractions during a display fade operation.

```
#define kCGDisplayBlendNormal (0.0)
#define kCGDisplayBlendSolidColor (1.0)
```

Constants
`kCGDisplayBlendNormal`
: Specifies that the blend color is not applied at the start or end of a fade operation.

  Available in Mac OS X v10.2 and later.

  Declared in `CGDisplayFade.h`.

`kCGDisplayBlendSolidColor`
: Specifies that the user sees only the blend color at the start or end of a fade operation.

  Available in Mac OS X v10.2 and later.

  Declared in `CGDisplayFade.h`.

Discussion
For general information about blend fractions, see the data type `CGDisplayBlendFraction` (page 85). For information about how these constants are used, see the function `CGDisplayFade` (page 33).

Display Fade Constants

Specifies values relating to fade operations.

```
#define kCGMaxDisplayReservationInterval (15.0)
#define kCGDisplayFadeReservationInvalidToken (0)
```

Constants
`kCGMaxDisplayReservationInterval`
: Specifies the maximum number of seconds for fade hardware reservations and display fade operations. For general information about fade intervals, see the data type `CGDisplayFadeInterval` (page 87).

  Available in Mac OS X v10.2 and later.

  Declared in `CGDisplayFade.h`.

`kCGDisplayFadeReservationInvalidToken`
Specifies an invalid fade reservation token. For general information about fade reservation tokens, see the data type `CGDisplayFadeReservationToken` (page 87).

Available in Mac OS X v10.2 and later.

Declared in `CGDisplayFade.h`.

Display ID Defaults

Default values for a display ID.

```
#define kCGDirectMainDisplay (CGMainDisplayID())
#define kCGNullDirectDisplay ((CGDirectDisplayID)0)
```

Constants

`kCGDirectMainDisplay`
Specifies the current main display ID.

Available in Mac OS X v10.0 and later.

Declared in `CGDirectDisplay.h`.

`kCGNullDirectDisplay`
Specifies a value that will never correspond to actual hardware.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

Display Mode Standard Properties

Specify keys for the standard properties in a display mode dictionary.

```
#define kCGDisplayWidth CFSTR("Width")
#define kCGDisplayHeight CFSTR("Height")
#define kCGDisplayMode CFSTR("Mode")
#define kCGDisplayBitsPerPixel CFSTR("BitsPerPixel")
#define kCGDisplayBitsPerSample CFSTR("BitsPerSample")
#define kCGDisplaySamplesPerPixel CFSTR("SamplesPerPixel")
#define kCGDisplayRefreshRate CFSTR("RefreshRate")
#define kCGDisplayModeUsableForDesktopGUICFSTR("UsableForDesktopGUI")
#define kCGDisplayIOFlags CFSTR("IOFlags")
#define kCGDisplayBytesPerRow CFSTR("kCGDisplayBytesPerRow")
```

Constants

`kCGDisplayWidth`
Specifies a CFNumber integer value that represents the width of the display in pixels.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayHeight`
Specifies a CFNumber integer value that represents the height of the display in pixels.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayMode`
Specifies a CFNumber integer value that represents the I/O Kit display mode number.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayBitsPerPixel`
Specifies a CFNumber integer value that represents the number of bits in a pixel.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayBitsPerSample`
Specifies a CFNumber integer value that represents the number of bits in an individual sample (for example, a color value in an RGB pixel).

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplaySamplesPerPixel`
Specifies a CFNumber integer value that represents the number of samples in a pixel.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayRefreshRate`
Specifies a CFNumber double-precision floating point value that represents the refresh rate of a CRT display. Some displays may not use conventional video vertical and horizontal sweep in painting the screen; these displays report a refresh rate of 0.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayModeUsableForDesktopGUI`
Specifies a CFBoolean value that indicates whether the display is suitable for use with the Mac OS X graphical user interface. The criteria include factors such as sufficient width and height and adequate pixel depth.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayIOFlags`
Specifies a CFNumber integer value that contains the I/O Kit display mode flags. For more information, see the header file `IOKit/IOGraphicsTypes.h`.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayBytesPerRow`
Specifies a CFNumber integer value that represents the number of bytes in a row on the display.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

Discussion
To learn how to use these keys to access the values in a display mode dictionary, see *CFDictionary Reference*.

Display Mode Optional Properties

Specify keys for optional properties in a display mode dictionary.

```
#define kCGDisplayModeIsSafeForHardware CFSTR ("kCGDisplayModeIsSafeForHardware")
#define kCGDisplayModeIsInterlaced CFSTR("kCGDisplayModeIsInterlaced")
#define kCGDisplayModeIsStretched CFSTR("kCGDisplayModeIsStretched")
#define kCGDisplayModeIsTelevisionOutput CFSTR ("kCGDisplayModeIsTelevisionOutput")
```

Constants

`kCGDisplayModeIsSafeForHardware`

Specifies a CFBoolean value indicating that the display mode doesn't need a confirmation dialog to be set.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayModeIsInterlaced`

Specifies a CFBoolean value indicating that the I/O Kit interlace mode flag is set.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayModeIsStretched`

Specifies a CFBoolean value indicating that the I/O Kit stretched mode flag is set.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayModeIsTelevisionOutput`

Specifies a CFBoolean value indicating that the I/O Kit television output mode flag is set.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

Discussion

A given key is present in a display mode dictionary only if the property applies, and is always associated with a value of `kCFBooleanTrue`. Keys not relevant to a particular display mode will not appear in the mode dictionary.

Reserved Window Levels

Specifies window level constants.

```
#define kCGNumReservedWindowLevels      (16)
```

Constants

`kCGNumReservedWindowLevels`

The number of window levels reserved by Apple for internal use. Application frameworks such as Carbon and Cocoa use this constant during compilation

Available in Mac OS X v10.0 and later.

Declared in `CGWindowLevel.h`.

Screen Update Operations

Specify types of screen update operations.

```
enum _CGScreenUpdateOperation {
    kCGScreenUpdateOperationRefresh = 0,
    kCGScreenUpdateOperationMove = (1 << 0),
    kCGScreenUpdateOperationReducedDirtyRectangleCount = (1 <<  31)
};
typedef uint32_t CGScreenUpdateOperation;
```

Constants

`kCGScreenUpdateOperationRefresh`

Specifies a screen refresh operation.

Available in Mac OS X v10.3 and later.

Declared in `CGRemoteOperation.h`.

`kCGScreenUpdateOperationMove`

Specifies a screen move operation.

Available in Mac OS X v10.3 and later.

Declared in `CGRemoteOperation.h`.

`kCGScreenUpdateOperationReducedDirtyRectangleCount`

When presented as part of the requested operations to the function `CGWaitForScreenUpdateRects` (page 76), specifies that the function should try to minimize the number of rectangles returned to represent the changed areas of the display. The function may combine adjacent rectangles within a larger bounding rectangle, which may include unmodified areas of the display.

Available in Mac OS X v10.4 and later.

Declared in `CGRemoteOperation.h`.

Discussion

For information about how these constants are used, see the function `CGWaitForScreenUpdateRects` (page 76).

Window Level Keys

Keys that represent the standard window levels in Mac OS X. Quartz includes these keys to support application frameworks such as Carbon and Cocoa. Applications do not need to use them directly.

```
enum _CGCommonWindowLevelKey {
 kCGBaseWindowLevelKey = 0,
 kCGMinimumWindowLevelKey,
 kCGDesktopWindowLevelKey,
 kCGBackstopMenuLevelKey,
 kCGNormalWindowLevelKey,
 kCGFloatingWindowLevelKey,
 kCGTornOffMenuWindowLevelKey,
 kCGDockWindowLevelKey,
 kCGMainMenuWindowLevelKey,
 kCGStatusWindowLevelKey,
 kCGModalPanelWindowLevelKey,
 kCGPopUpMenuWindowLevelKey,
 kCGDraggingWindowLevelKey,
 kCGScreenSaverWindowLevelKey,
 kCGMaximumWindowLevelKey,
 kCGOverlayWindowLevelKey,
 kCGHelpWindowLevelKey,
 kCGUtilityWindowLevelKey,
 kCGDesktopIconWindowLevelKey,
 kCGCursorWindowLevelKey,
 kCGNumberOfWindowLevelKeys
};
typedef int32_t CGWindowLevelKey;
```

Constants

`kCGBaseWindowLevelKey`
: The base key used to define window levels. Do not use.

  Available in Mac OS X v10.0 and later.

  Declared in `CGWindowLevel.h`.

`kCGMinimumWindowLevelKey`
: The lowest available window level.

  Available in Mac OS X v10.0 and later.

  Declared in `CGWindowLevel.h`.

`kCGDesktopWindowLevelKey`
: The level for the desktop.

  Available in Mac OS X v10.0 and later.

  Declared in `CGWindowLevel.h`.

`kCGBackstopMenuLevelKey`
: The level of the backstop menu.

  Available in Mac OS X v10.0 and later.

  Declared in `CGWindowLevel.h`.

`kCGNormalWindowLevelKey`
: The level for normal windows.

  Available in Mac OS X v10.0 and later.

  Declared in `CGWindowLevel.h`.

`kCGFloatingWindowLevelKey`
: The level for floating windows.

  Available in Mac OS X v10.0 and later.

  Declared in `CGWindowLevel.h`.

`kCGTornOffMenuWindowLevelKey`
The level for torn off menus.

Available in Mac OS X v10.0 and later.

Declared in `CGWindowLevel.h`.

`kCGDockWindowLevelKey`
The level for the dock.

Available in Mac OS X v10.0 and later.

Declared in `CGWindowLevel.h`.

`kCGMainMenuWindowLevelKey`
The level for the menus displayed in the menu bar.

Available in Mac OS X v10.0 and later.

Declared in `CGWindowLevel.h`.

`kCGStatusWindowLevelKey`
The level for status windows.

Available in Mac OS X v10.0 and later.

Declared in `CGWindowLevel.h`.

`kCGModalPanelWindowLevelKey`
The level for modal panels.

Available in Mac OS X v10.0 and later.

Declared in `CGWindowLevel.h`.

`kCGPopUpMenuWindowLevelKey`
The level for pop-up menus.

Available in Mac OS X v10.0 and later.

Declared in `CGWindowLevel.h`.

`kCGDraggingWindowLevelKey`
The level for a window being dragged.

Available in Mac OS X v10.0 and later.

Declared in `CGWindowLevel.h`.

`kCGScreenSaverWindowLevelKey`
The level for screen savers.

Available in Mac OS X v10.0 and later.

Declared in `CGWindowLevel.h`.

`kCGMaximumWindowLevelKey`
The highest allowed window level.

Available in Mac OS X v10.0 and later.

Declared in `CGWindowLevel.h`.

`kCGOverlayWindowLevelKey`
The level for overlay windows.

Available in Mac OS X v10.1 and later.

Declared in `CGWindowLevel.h`.

`kCGHelpWindowLevelKey`
The level for help windows.

Available in Mac OS X v10.1 and later.

Declared in `CGWindowLevel.h`.

`kCGUtilityWindowLevelKey`
The level for utility windows.

Available in Mac OS X v10.1 and later.

Declared in `CGWindowLevel.h`.

`kCGDesktopIconWindowLevelKey`
The level for desktop icons.

Available in Mac OS X v10.1 and later.

Declared in `CGWindowLevel.h`.

`kCGCursorWindowLevelKey`
The level for the cursor.

Available in Mac OS X v10.2 and later.

Declared in `CGWindowLevel.h`.

`kCGNumberOfWindowLevelKeys`
The total number of window levels.

Available in Mac OS X v10.0 and later.

Declared in `CGWindowLevel.h`.

Window Server Session Properties

Specify keys for the standard properties in a window server session dictionary.

```
#define kCGSessionUserIDKey CFSTR("kCGSSessionUserIDKey")
#define kCGSessionUserNameKey CFSTR("kCGSSessionUserNameKey")
#define kCGSessionConsoleSetKey CFSTR("kCGSSessionConsoleSetKey")
#define kCGSessionOnConsoleKey CFSTR("kCGSSessionOnConsoleKey")
#define kCGSessionLoginDoneKey CFSTR("kCGSessionLoginDoneKey")
```

Constants

`kCGSessionUserIDKey`
Specifies a CFNumber 32-bit unsigned integer value that encodes a user ID for the session's current user.

Available in Mac OS X v10.3 and later.

Declared in `CGSession.h`.

`kCGSessionUserNameKey`
Specifies a CFString value that encodes the session's short user name as set by the login operation.

Available in Mac OS X v10.3 and later.

Declared in `CGSession.h`.

`kCGSessionConsoleSetKey`
Specifies a CFNumber 32-bit unsigned integer value that represents a set of hardware composing a console.

Available in Mac OS X v10.3 and later.

Declared in `CGSession.h`.

`kCGSessionOnConsoleKey`
Specifies a CFBoolean value indicating whether the session is on a console.

Available in Mac OS X v10.3 and later.

Declared in `CGSession.h`.

`kCGSessionLoginDoneKey`
Specifies a CFBoolean value indicating whether the login operation has been done.

Available in Mac OS X v10.3 and later.

Declared in `CGSession.h`.

Discussion
To learn how to use these keys to access the values in a session dictionary, see *CFDictionary Reference*.

Result Codes

This table lists the result codes returned by functions in Quartz Display Services.

Result Code	Value	Description
`kCGErrorSuccess`	0	The requested operation was completed successfully. Available in Mac OS X v10.0 and later.
`kCGErrorFailure`	1000	A general failure occurred. Available in Mac OS X v10.0 and later.
`kCGErrorIllegalArgument`	1001	One or more of the parameters passed to a function are invalid. Check for `NULL` pointers. Available in Mac OS X v10.0 and later.
`kCGErrorInvalidConnection`	1002	The parameter representing a connection to the window server is invalid. Available in Mac OS X v10.0 and later.
`kCGErrorInvalidContext`	1003	The `CPSProcessSerNum` or context identifier parameter is not valid. Available in Mac OS X v10.0 and later.
`kCGErrorCannotComplete`	1004	The requested operation is inappropriate for the parameters passed in, or the current system state. Available in Mac OS X v10.0 and later.

Result Code	Value	Description
`kCGErrorNameTooLong`	1005	A parameter, typically a C string, is too long to be used without truncation. Available in Mac OS X v10.0 and later.
`kCGErrorNotImplemented`	1006	Return value from obsolete function stubs present for binary compatibility, but not normally called. Available in Mac OS X v10.0 and later.
`kCGErrorRangeCheck`	1007	A parameter passed in has a value that is inappropriate, or which does not map to a useful operation or value. Available in Mac OS X v10.0 and later.
`kCGErrorTypeCheck`	1008	A data type or token was encountered that did not match the expected type or token. Available in Mac OS X v10.0 and later.
`kCGErrorNoCurrentPoint`	1009	An operation relative to a known point or coordinate could not be done, as there is no known point. Available in Mac OS X v10.0 and later.
`kCGErrorInvalidOperation`	1010	The requested operation is not valid for the parameters passed in, or the current system state. Available in Mac OS X v10.0 and later.
`kCGErrorNoneAvailable`	1011	The requested operation could not be completed as the indicated resources were not found. Available in Mac OS X v10.0 and later.

Document Revision History

This table describes the changes to *Quartz Display Services Reference*.

Date	Notes
2008-11-19	Updated the description of the CGGetDisplaysWithRect function.
2007-10-31	Updated for Mac OS X v10.5.
	Added descriptions of the functions `CGDisplayCopyColorSpace` (page 32) and `CGDisplayRotation` (page 46).
2006-09-20	Made minor format changes.
	Added information about mirrored displays to `CGConfigureDisplayMirrorOfDisplay` (page 19), `CGDisplayCapture` (page 31) and `CGGetActiveDisplayList` (page 53).
2006-06-28	Made minor technical corrections.
	Updated descriptions of the functions `CGConfigureDisplayMode` (page 20), `CGDisplaySwitchToMode` (page 50), and `CGShieldingWindowLevel` (page 74).
2006-04-04	Added descriptions for previously undocumented functions and constants, and changed the title from "Quartz Services Reference."
2005-08-11	Added information about registering for notification of display configuration changes.
2003-11-13	Released as a preliminary document.

Document Revision History

Index

C

D

K

R

S

W