

---

# QuickTime Data Types Reference

QuickTime



2006-05-23



Apple Inc.  
© 2006 Apple Computer, Inc.  
All rights reserved.

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

The Apple logo is a trademark of Apple Inc.

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

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

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

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

.Mac is a registered service mark of Apple Inc.

Apple, the Apple logo, Mac, Mac OS, Macintosh, QuickDraw, and QuickTime are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

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

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

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

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

# Contents

## QuickTime Data Types Reference 7

---

|                         |    |
|-------------------------|----|
| Overview                | 7  |
| Data Types              | 7  |
| CallbackRecord          | 7  |
| CGrafPort               | 8  |
| CodecInfo               | 12 |
| ComponentInstanceRecord | 17 |
| EventRecord             | 17 |
| FixedPoint              | 18 |
| FSSpec                  | 19 |
| ICMAlignmentProcRecord  | 19 |
| ICMCompletionProcRecord | 20 |
| ICMDataProcRecord       | 21 |
| ICMFlushProcRecord      | 22 |
| ICMFrameTimeRecord      | 22 |
| ICMProgressProcRecord   | 24 |
| MatrixRecord            | 25 |
| MediaRecord             | 25 |
| MovieRecord             | 25 |
| PixMap                  | 26 |
| Point                   | 28 |
| QTEventRecord           | 29 |
| Rect                    | 30 |
| RGBColor                | 30 |
| TimeBaseRecord          | 31 |
| TimeRecord              | 31 |
| ToneDescription         | 32 |
| TrackRecord             | 33 |
| UserDataRecord          | 34 |
| wide                    | 34 |
| ActionsUPP              | 34 |
| AliasHandle             | 35 |
| AliasPtr                | 35 |
| ByteCount               | 35 |
| CGrafPtr                | 35 |
| CodecQ                  | 36 |
| CodecType               | 36 |
| ComponentInstance       | 36 |
| ComponentResult         | 36 |
| CompressorComponent     | 37 |
| ConstStr255Param        | 37 |

|                            |    |
|----------------------------|----|
| CTabHandle                 | 37 |
| CTabPtr                    | 37 |
| DataHandler                | 37 |
| DialogPtr                  | 38 |
| DialogRef                  | 38 |
| DoMCActionUPP              | 38 |
| GDHandle                   | 38 |
| GDPtr                      | 39 |
| GWorldFlags                | 39 |
| GWorldPtr                  | 39 |
| ICMAlignmentProcRecordPtr  | 39 |
| ICMCompletionProcRecordPtr | 40 |
| ICMConvertDataFormatUPP    | 40 |
| ICMDataProcRecordPtr       | 40 |
| ICMFlushProcRecordPtr      | 40 |
| ICMMemoryDisposedUPP       | 40 |
| ICMProgressProcRecordPtr   | 41 |
| ImageDescriptionHandle     | 41 |
| ImageDescriptionPtr        | 41 |
| ImageSequence              | 41 |
| ItemCount                  | 42 |
| MatrixRecordPtr            | 42 |
| Media                      | 42 |
| MediaHandler               | 42 |
| MenuHandle                 | 43 |
| MenuRef                    | 43 |
| ModalFilterUPP             | 43 |
| Movie                      | 43 |
| MovieController            | 43 |
| MovieDrawingCompleteUPP    | 44 |
| MoviePrePrerollCompleteUPP | 44 |
| MoviePreviewCallOutUPP     | 44 |
| MovieProgressUPP           | 44 |
| MoviesErrorUPP             | 45 |
| OSErr                      | 45 |
| OSStatus                   | 45 |
| PicHandle                  | 45 |
| PicPtr                     | 46 |
| PixMapHandle               | 46 |
| PixMapPtr                  | 46 |
| QTAtom                     | 46 |
| QTAtomContainer            | 46 |
| QTAtomID                   | 47 |
| QTCallback                 | 47 |
| QTCallbackUPP              | 47 |
| QTEventRecordPtr           | 47 |

|                                   |    |
|-----------------------------------|----|
| QTNextTaskNeededSoonerCallbackUPP | 48 |
| QTParameterDialog                 | 48 |
| QTParameterDialogOptions          | 48 |
| RgnHandle                         | 48 |
| RgnPtr                            | 49 |
| SampleDescriptionHandle           | 49 |
| SampleDescriptionPtr              | 49 |
| ScriptCode                        | 49 |
| Size                              | 49 |
| SoundDescriptionHandle            | 50 |
| SoundDescriptionPtr               | 50 |
| Str255                            | 50 |
| StringPtr                         | 50 |
| TextMediaUPP                      | 51 |
| TimeBase                          | 51 |
| TimeScale                         | 51 |
| TimeValue                         | 51 |
| TimeValue64                       | 52 |
| Track                             | 52 |
| TrackTransferUPP                  | 52 |
| UserData                          | 52 |
| VdigIntUPP                        | 52 |
| WindowPtr                         | 53 |
| WindowRef                         | 53 |

---

## Document Revision History 55

---

## Index 57

---

# C O N T E N T S

# QuickTime Data Types Reference

---

|                    |   |
|--------------------|---|
| <b>Framework:</b>  | Frameworks/QuickTime.framework  |
| <b>Declared in</b> | Aliases.h<br>Components.h<br>Dialogs.h<br>IOHIDDescriptorParser.h<br>IOMacOSTypes.h<br>ImageCompression.h<br>MacTypes.h<br>Menus.h<br>Movies.h<br>OSTypes.h<br>QDOffscreen.h<br>QuickTimeComponents.h<br>QuickdrawTypes.h |

## Overview

This reference covers the data types common to multiple QuickTime frameworks.

## Data Types

### CallbackRecord

Stores data for a QTCallbackProc.

```
struct CallbackRecord {  
    long    data[1];  
};
```

#### Fields

data

#### Discussion

Callback data.

#### Programming Info

C interface file: Movies.h

**CGrafPort**

Defines a complete drawing environment for color graphics operations.

```
struct CGrafPort {
    short          device;
    PixMapHandle   portPixMap;
    short          portVersion;
    Handle         grafVars;
    short          chExtra;
    short          pnLocHFrac;
    Rect           portRect;
    RgnHandle      visRgn;
    RgnHandle      clipRgn;
    PixPatHandle   bkPixPat;
    RGBColor       rgbFgColor;
    RGBColor       rgbBkColor;
    Point          pnLoc;
    Point          pnSize;
    short          pnMode;
    PixPatHandle   pnPixPat;
    PixPatHandle   fillPixPat;
    short          pnVis;
    short          txFont;
    StyleField     txFace;
    short          txMode;
    short          txSize;
    Fixed          spExtra;
    long           fgColor;
    long           bkColor;
    short          colrBit;
    short          patStretch;
    Handle         picSave;
    Handle         rgnSave;
    Handle         polySave;
    CQDProcsPtr    grafProcs;
};
```

**Fields**

device

**Discussion**

Device-specific information that QuickDraw uses to achieve the best possible results when drawing text in the graphics port. There may be physical differences in the same logical font for different output devices, to ensure the highest-quality printing on the device being used. The default value of the `device` field is 0, indicating the computer screen.

portPixMap

**Discussion**

A handle to a `PixMap` structure, which describes the pixels in this color graphics port.

portVersion

**Discussion**

The highest 2 bits are permanently set to indicate that this is a `CGrafPort` structure and the remainder of the field contains the version number of Macintosh Color QuickDraw that created this structure. Currently initialized to 0xC000.



grafVars

**Discussion**

A handle to a `GrafVars` structure that contains additional graphics fields of color information. On initialization, black is assigned to the `rgbOpColor` field of this structure, the default highlight color is assigned to the `rgbHiLiteColor` field, and all other fields are set to 0. For information about the `GrafVars` structure, see *Inside Macintosh: Imaging With QuickDraw*.

chExtra

**Discussion**

A number by which to widen every character, excluding spaces, in a line of text. This value is used in proportional spacing. The value in this field is in 4.12 fractional notation: 4 bits of signed integer followed by 12 bits of fraction. This value is multiplied by the value in the `txSize` field before it is used. By default, this field contains 0.

pnLocHFrac

**Discussion**

The fractional horizontal pen position used when drawing text. The value in this field represents the low word of type `Fixed`; in decimal, its initial value is 0.5.

portRect

**Discussion**

The port rectangle that defines a subset of the pixel map to be used for drawing. All drawing done by the application occurs inside the port rectangle. The port rectangle (also called the content region) uses the local coordinate system defined by the boundary rectangle in the `portPixMap` field of the `PixMap` structure. The upper-left corner (which for a window is called the window origin) of the port rectangle usually has vertical and horizontal coordinates of 0. The port rectangle usually falls within the boundary rectangle, but it's not required to do so.

visRgn

**Discussion**

The region of the graphics port that's actually visible on the screen; that is, the part of the window that's not covered by other windows. By default, the visible region is equivalent to the port rectangle. The visible region has no effect on images that aren't displayed on the screen.

clipRgn

**Discussion**

A handle to the graphics port's clipping region, an arbitrary region that you can use to limit drawing to any region within the port rectangle. Unlike the visible region, the clipping region affects the image even if it isn't displayed on the screen. Initially the clip region is set to the rectangle -32768, -32768, 32767, 32767.

bkPixPat

**Discussion**

A handle to a `PixPat` structure that describes the background pixel pattern, initially set to white.

rgbFgColor

**Discussion**

An `RGBColor` structure that defines the requested foreground color. By default, the foreground color is black.

rgbBkColor

**Discussion**

An `RGBColor` structure that defines the requested background color. By default, the background color is white.

`pnLoc`

**Discussion**

The point where QuickTime will begin drawing the next line, shape, or character. It can be anywhere on the coordinate plane; there are no restrictions on the movement or placement of the pen. The location of the graphics pen is a point in the graphics port's coordinate system, not a pixel in a pixel image. The upper-left corner of the pen is at the pen location; the graphics pen hangs below and to the right of this point. This field is initialized to 0,0.

`pnSize`

**Discussion**

The vertical height and horizontal width of the graphics pen. The default size is a 1-by-1 pixel square; the vertical height and horizontal width can range from 0 by 0 to 32,767 by 32,767. If either the pen width or the pen height is 0, the pen does not draw. Heights or widths of less than 0 are undefined.

`pnMode`

**Discussion**

The pattern mode, a Boolean operation that determines the how QuickTime transfers the pen pattern to the pixel map during drawing operations. See `Graphics Transfer Modes`. When the graphics pen draws into a pixel map, QuickTime first determines what pixels in the pixel image are affected and finds their corresponding pixels in the pen pattern. It then does a pixel-by-pixel comparison based on the pattern mode, which specifies one of eight Boolean transfer operations to perform. QuickTime stores the resulting pixel in its proper place in the image. This field is initially set to `patCopy`.

`pnPixPat`

**Discussion**

A handle to a `PixPat` structure that describes a pixel pattern that can be used like the ink in the graphics pen. This field is initially set to black.

`fillPixPat`

**Discussion**

A handle to a `PixPat` structure that describes the pixel pattern that's used to fill an area. This field is initially set to black. Notice that this is not in the same location as the `fillPat` field in the `GrafPort` structure.

`pnVis`

**Discussion**

The graphics pen's visibility; that is, whether or not it draws on the screen. This field is initially set to 0 (visible).

`txFont`

**Discussion**

A font number that identifies the font to be used in the graphics port. This field is initially set to 0, indicating the system font.

`txFace`

**Discussion**

The character `style` of the text, with values from the set defined by the `Style` type, which includes such styles as bold, italic, and shaded. You can apply stylistic variations either alone or in combination. This field is initially set to plain text.

`txMode`

**Discussion**

One of three Boolean source mode constants (see below) that determines the way characters are placed in the bit image. This mode functions much like a pattern mode specified in the `pnMode` field; when drawing a character, QuickTime determines which pixels in the image are affected, does a pixel-by-pixel comparison based on the mode, and stores the resulting pixels in the image. This field is initially set to `srcOr`. See these constants:

`txSize`

**Discussion**

The text size in pixels. QuickTime uses this information to provide the bitmaps for text drawing. The `txSize` value can be represented by the formula (size in points) x (device resolution) / 72 dpi. This field is initially set to the system font size.

`spExtra`

**Discussion**

A number equal to the average number of pixels by which each space character should be widened to fill out a fully justified text line. This field is useful when a line of characters is to be aligned with both the left and the `right` margin. This field is initially set to 0.

`fgColor`

**Discussion**

The pixel value of the foreground color. This is the best available approximation in the color lookup table (CLUT) to the color specified in the `rgbFgColor` field. This field is initially set to `blackColor`; see `Color Constants`.

`bkColor`

**Discussion**

The pixel value of the background color. This is the best available approximation in the color lookup table (CLUT) to the color specified in the `rgbBkColor` field. This field is initially set to `whiteColor`; see `Color Constants`.

`colrBit`

**Discussion**

Reserved and set to 0.

`patStretch`

**Discussion**

A value, initially set to 0, used during output to a printer to expand patterns if necessary. Your application should not change this value.

`picSave`

**Discussion**

A handle to the state of the Macintosh picture definition. If no picture is open, this field contains `NIL`; otherwise it contains a handle to information related to the picture definition. Your application shouldn't be concerned about exactly what information the handle leads to; you may, however, save the current value of this field, set the field to `NIL` to disable the picture definition, and later restore it to the saved value to resume defining the picture.

`rgnSave`

#### **Discussion**

A handle to the state of the region definition. If no region is open, this field contains `NIL`; otherwise it contains a handle to information related to the region definition. Your application shouldn't be concerned about exactly what information the handle leads to; you may, however, save the current value of this field, set the field to `NIL` to disable the region definition, and later restore it to the saved value to resume defining the region.

`polySave`

#### **Discussion**

A handle to the state of the polygon definition. If no polygon is open, this field contains `NIL`; otherwise it contains a handle to information related to the polygon definition. Your application shouldn't be concerned about exactly what information the handle leads to; you may, however, save the current value of this field, set the field to `NIL` to disable the polygon definition, and later restore it to the saved value to resume defining the polygon.

`grafProcs`

#### **Discussion**

An optional pointer to a `CQDProcs` structure that your application can store into if you want to customize Color QuickDraw drawing routines or use Color QuickDraw in other advanced, highly specialized ways. This field is initially set to `NIL`.

#### **Discussion**

You can have many graphics ports open at once; each one has its own local coordinate system, pen pattern, background pattern, pen size and location, font and font `style`, and pixel map in which drawing takes place. Several fields in this structure define your application's drawing area. All drawing in a graphics port occurs in the intersection of the graphics port's boundary rectangle and its port rectangle. Within that intersection, all drawing is cropped to the graphics port's visible region and its clipping region.

#### **Version Notes**

The `CGrafPort` structure supersedes the earlier `GrafPort` structure.

#### **Programming Info**

C interface file: `Quickdraw.h`

## **CodeInfo**

Describes the capabilities of a compressor.

```

struct CodecInfo {
    Str31          typeName;
    short          version;
    short          revisionLevel;
    long           vendor;
    long           decompressFlags;
    long           compressFlags;
    long           formatFlags;
    UInt8          compressionAccuracy;
    UInt8          decompressionAccuracy;
    unsigned short compressionSpeed;
    unsigned short decompressionSpeed;
    UInt8          compressionLevel;
    UInt8          resvd;
    short          minimumHeight;
    short          minimumWidth;
    short          decompressPipelineLatency;
    short          compressPipelineLatency;
    long           privateData;
};

```

**Fields**

typeName

**Discussion**

Indicates the compression algorithm used by the component; for example, 'Animation'. This Pascal string may be used to identify the compression algorithm to the user. The string always takes up 32 bytes no matter how long it is. The 32 bytes consist of 31 bytes plus one length byte. Apple assigns these type names. The value of this field should correspond to the value of the typeName field in the appropriate compressor name structure returned by `GetCodecNameList`.

version

**Discussion**

Indicates the version of compressed data this component supports. The contents of this field should indicate the most recent version of the compression algorithm that the component can understand.

revisionLevel

**Discussion**

Indicates the version of the component; for example, 0x00010001 (1.0.1). Developers of compressors assign these version numbers.

vendor

**Discussion**

Identifies the developer of the component; for example, 'appl'. The value of this field corresponds to the manufacturer code or application signature assigned to the developer.

`decompressFlags`

### Discussion

Contains flags (see below) that specify the decompression capabilities of the component. Typically, these flags are of interest only to developers of image decompressors. See these constants:

- `codecInfoDoes1`
- `codecInfoDoes2`
- `codecInfoDoes4`
- `codecInfoDoes8`
- `codecInfoDoes16`
- `codecInfoDoes32`
- `codecInfoDoesDither`
- `codecInfoDoesStretch`
- `codecInfoDoesShrink`
- `codecInfoDoesMask`
- `codecInfoDoesTemporal`
- `codecInfoDoesDouble`
- `codecInfoDoesQuad`
- `codecInfoDoesHalf`
- `codecInfoDoesQuarter`
- `codecInfoDoesRotate`
- `codecInfoDoesHorizFlip`
- `codecInfoDoesVertFlip`
- `codecInfoHasEffectParameterList`
- `codecInfoDoesBlend`
- `codecInfoDoesWarp`
- `codecInfoDoesRecompress`
- `codecInfoDoesSpool`
- `codecInfoDoesRateConstrain`

`compressFlags`

### Discussion

Contains flags (see below) that specify the compression capabilities of the component. Typically, these flags are of interest only to developers of image compressors.

`formatFlags`

#### Discussion

Contains flags (see below) that describe the possible format for compressed data produced by this component and the format of compressed files that the component can handle during decompression. Typically, these flags are of interest only to developers of compressor components. See these constants:

```

codecInfoDepth1
codecInfoDepth2
codecInfoDepth4
codecInfoDepth8
codecInfoDepth16
codecInfoDepth24
codecInfoDepth32
codecInfoDepth33
codecInfoDepth34
codecInfoDepth36
codecInfoDepth40
codecInfoStoresClut
codecInfoDoesLossless
codecInfoSequenceSensitive

```

`compressionAccuracy`

#### Discussion

Indicates the relative accuracy of the compression algorithm employed by the component. Valid values for this field range from 0 to 255. A value of 0 means that the accuracy is unknown. Values from 1 to 255 provide a gauge for the relative accuracy of the compression algorithm; higher values indicate better accuracy. The Image Compression Manager examines this field to determine which compressor component can most accurately compress a given image. The `compressionAccuracy` field can only approximate the accuracy of a compression algorithm. Typically, compression algorithms produce results of varying quality based on a variety of parameters, including image size and content. Since this information is not available until a compression request is issued, a precise measure of accuracy is not possible. However, the `value` of this field should still give a rough idea of the accuracy of the supported algorithm.

`decompressionAccuracy`

#### Discussion

Indicates the relative accuracy of the decompression algorithm employed by the component. Valid values for this field range from 0 to 255. A value of 0 means that the accuracy is unknown. Values from 1 to 255 indicate the relative accuracy of the decompression technique; higher values mean better accuracy. The Image Compression Manager examines this field to determine which decompressor component can most accurately decompress a given image. The `decompressionAccuracy` field can only approximate the accuracy of a decompression algorithm. Typically, decompression algorithms produce results of varying quality based on a variety of parameters, including image size and content. Since this information is not available until a decompression request is issued, a precise measure of accuracy is not possible. However, the `value` of this field should still give a rough idea of the accuracy of the supported algorithm.

`compressionSpeed`

#### Discussion

Indicates the relative speed of the component for compression operations. Valid values for this field lie in the range from 0 to 65,535. A value of 0 means that the speed is unknown. Values from 1 to 65,535 correspond to the number of milliseconds the component requires to compress a 320-by-240 pixel image on a Macintosh II computer. The Image Compression Manager examines this field to determine which compressor component can most quickly compress a given image.

`decompressionSpeed`

**Discussion**

Indicates the relative speed of the component for decompression operations. Valid values for this field lie in the range from 0 to 65,535. A value of 0 means that the speed is unknown. Values from 1 to 65,535 correspond to the number of milliseconds the component requires to decompress a 320-by-240 pixel image on a Macintosh II computer. The Image Compression Manager examines this field to determine which compressor component can most quickly decompress a given image.

`compressionLevel`

**Discussion**

Indicates the relative compression achieved by this component. Valid values for this field lie in the range from 0 to 255. A value of 0 means that the compression level is unknown. Values from 1 to 255 map to percentage values of relative compression; lower values mean lesser compression. A value of 1 means no compression (0 percent); a value of 255 means maximum compression (100 percent). The Image Compression Manager examines this field to determine which available compressor component will yield the smallest resulting data for a given image. The `compressionLevel` field can only approximate the effectiveness of a compression algorithm. Typically, compression algorithms produce results of varying quality based on a variety of parameters, including image size and content. Since this information is not available until a compression request is issued, a precise measure of compression is not possible. However, the `value` of this field should still give a rough idea of the effectiveness of the supported algorithm.

`resvd`

**Discussion**

Reserved; set to 0.

`minimumHeight`

**Discussion**

Specifies the height in pixels of the smallest image the component can handle. Together with the `minimumWidth` field, this field defines the block size for the component. The Image Compression Manager does not issue compression or decompression requests for images smaller than the block size.

`minimumWidth`

**Discussion**

Specifies the width in pixels of the smallest image the component can handle. Together with the `minimumHeight` field, this field defines the block size for the component. The Image Compression Manager does not issue compression or decompression requests for images smaller than the block size.

`decompressPipelineLatency`

**Discussion**

Decompression pipeline latency in milliseconds, for asynchronous codecs.

`compressPipelineLatency`

**Discussion**

Compression pipeline latency in milliseconds, for asynchronous codecs.

`privateData`

**Discussion**

Reserved for future use. This field must be set to 0.

**Discussion**

Contains the description of a codec.

**Version Notes**

The `codecInfoHasEffectParameterList` constant was formerly `codecInfoDoesSkew`.



#### Related Functions

GetCodecInfo

ImageCodecGetCodecInfo

#### Programming Info

C interface file: ImageCompression.h

## ComponentInstanceRecord

Undocumented

```
struct ComponentInstanceRecord {
    long    data[1];
};
```

#### Fields

data

#### Discussion

*Undocumented*

#### Discussion

*Undocumented*

#### Programming Info

C interface file: Components.h

## EventRecord

Contains information about a retrieved Mac OS event.

```
struct EventRecord {
    EventKind    what;
    UInt32       message;
    UInt32       when;
    Point        where;
    EventModifiers modifiers;
};
```

#### Fields

what

#### Discussion

A constant (see below) that specifies the kind of event. See these constants:

message

#### Discussion

Additional information (see below) associated with the event. The interpretation of this information depends on the event type. See these constants:

when

#### Discussion

The time when the event was posted, in ticks since system startup.

where

#### Discussion

For low-level events and operating-system events, this field contains the location of the cursor at the time the event was posted (in global coordinates). For high-level events, it contains a second event specifier, the event ID. The event ID defines the particular type of event within the class of events defined by the `message` field of the high-level event. For high-level events, you should interpret the `where` field as having the data type `OSType`, not `Point`.

modifiers

#### Discussion

Contains information about the state of the modifier keys and the mouse button at the time the event was posted. For activate events, this field also indicates whether the window should be activated or deactivated. In System 7 it also indicates whether the mouse-down event caused your application to switch to the foreground. Each modifier key is represented by a specific bit in the `modifiers` field of the event record structure. The modifier keys include the Option, Command, Caps Lock, Control, and Shift keys. If your application attaches special meaning to any of these keys in combination with other keys or when the mouse button is down, you can test the state of the `modifiers` field to determine the action your application should take. For example, you can use this information to determine whether the user pressed the Command key and another key to make a menu choice.

#### Related Functions

ImageCodecIsStandardParameterDialogEvent  
ModalFilterProc  
ModalFilterYDProc  
NativeEventToMacEvent  
PreviewEvent  
QTIsStandardParameterDialogEvent  
SCModalFilterProc  
SFModalFilterProc  
WinEventToMacEvent

#### Programming Info

C interface file: `Events.h`

## FixedPoint

Defines the position of a geometric point in fixed-point numbers.

```
struct FixedPoint {
    Fixed    x;
    Fixed    y;
};
```

#### Fields

x

#### Discussion

The x (horizontal) coordinate of the point.

y

#### Discussion

The y (vertical) coordinate of the point.

#### Related Functions

CurveGetNearestPathPoint  
CurveLengthToPoint  
CurvePathPointToLength  
TransformFixedPoints  
TransformFixedRect  
TransformRect

#### Programming Info

C interface file: `MacTypes.h`

## FSSpec

Identifies a Mac OS file or directory.

```
struct FSSpec {  
    short      vRefNum;  
    long       parID;  
    StrFileName name;  
};
```

#### Fields

`vRefNum`

#### Discussion

Volume reference number.

`parID`

#### Discussion

Directory ID of parent directory.

`name`

#### Discussion

Filename or directory name; a `Str63` string on the Mac OS.

#### Discussion

The `FSSpec` structure provides a simple and standard format for specifying files and directories. You can pass that specification directly to any file-manipulation routines that accept `FSSpec` records.

#### Related Functions

ConvertMovieToFile  
GraphicsExportGetInputFile  
GraphicsExportGetOutputFile  
GraphicsImportDoExportImageFileDialog  
GraphicsImportGetDataFile  
NativePathNameToFSSpec  
SGGetDataOutput

#### Programming Info

C interface file: `Files.h`

## ICMAlignmentProcRecord

Specifies an image compression alignment callback.

```
struct ICMAAlignmentProcRecord {
    ICMAAlignmentUPP    alignmentProc;
    long                alignmentRefCon;
};
```

**Fields**

alignmentProc

**Discussion**

Contains a Universal Procedure Pointer that accesses your ICMAAlignmentProc callback.

alignmentRefCon

**Discussion**

Contains a reference constant for use by your callback.

**Discussion**

This structure defines a pointer to an alignment function. You assign an alignment function by passing a pointer to this structure.

**Related Functions**

AlignScreenRect

AlignWindow

DragAlignedGrayRgn

DragAlignedWindow

SGGetAlignmentProc

**Programming Info**

C interface file: ImageCompression.h

**ICMCompletionProcRecord**

Specifies an image compression completion callback.

```
struct ICMCompletionProcRecord {
    ICMCompletionUPP    completionProc;
    long                completionRefCon;
};
```

**Fields**

completionProc

**Discussion**

Contains a Universal Procedure Pointer that accesses your ICMCompletionProc callback.

completionRefCon

**Discussion**

Contains a reference constant for use by your callback.

**Discussion**

This structure governs whether you perform a compression asynchronously. If the completionProc field in this structure is set to NIL, perform the compression synchronously. If this field is not NIL, it specifies an application completion function. Perform the compression asynchronously and call that completion function when your component is finished. If the completionProc field in this structure has a value of -1, perform the operation asynchronously but do not call the application's completion function

### Related Functions

CompressSequenceFrame  
DecompressSequenceFrame  
DecompressSequenceFrames  
DecompressSequenceFrameWhen  
ICMDecompressComplete  
ICMDecompressCompleteS  
MediaQueueNonPrimarySourceData  
MediaSetNonPrimarySourceData  
SCCompressSequenceFrameAsync  
TweenDataProc

### Programming Info

C interface file: `ImageCompression.h`

## ICMDataProcRecord

Specifies an image compression data-loading function.

```
struct ICMDataProcRecord {  
    ICMDataUPP      dataProc;  
    long            dataRefCon;  
};
```

### Fields

`dataProc`

### Discussion

Contains a pointer to your data-loading function.

`dataRefCon`

### Discussion

Contains a reference constant for use by your data-loading function.

### Discussion

If there is no data-loading function, the Image Compression Manager sets the `dataProc` field to `NIL`, and the entire image must be in memory at the location specified by the `codecData` field of the `ImageSubCodecDecompressRecord` structure.

### Related Functions

FDecompressImage  
GetCompressedImageSize  
GetCompressedPixMapInfo  
ImageCodecGetCompressedImageSize  
ImageCodecTrimImage  
SetCompressedPixMapInfo  
SetDSequenceDataProc  
TrimImage

### Programming Info

C interface file: `ImageCompression.h`

**ICMFlushProcRecord**

Specifies an image compression data-unloading callback.

```
struct ICMFlushProcRecord {
    ICMFlushUPP    flushProc;
    long           flushRefCon;
};
```

**Fields**

`flushProc`

**Discussion**

Contains a pointer to your data-unloading function.

`flushRefCon`

**Discussion**

Contains a reference constant for use by your data-unloading function.

**Discussion**

If there is not enough memory to store a compressed image, your application may provide a function that unloads some of the compressed data. This field contains a structure that identifies that data-unloading function. If the application did not provide a data-unloading function, the `flushProc` field in this structure is set to `NIL`. In this case, your component writes the entire compressed image into the memory location specified by the `data` field.

**Related Functions**

`FCompressImage`

`ImageCodecTrimImage`

`SetCSequenceFlushProc`

`TrimImage`

**Programming Info**

C interface file: `ImageCompression.h`

**ICMFrameTimeRecord**

Contains a frame's time information for scheduled asynchronous decompression operations.

```

struct ICMFrameTimeRecord {
    wide        value;
    long        scale;
    void *      base;
    long        duration;
    Fixed       rate;
    long        recordSize;
    long        frameNumber;
    long        flags;
    wide        virtualStartTime;
    long        virtualDuration;
    TimeValue64 decodeTime;
};

```

**Fields**

value

**Discussion**

Specifies the time at which the frame is to be displayed.

scale

**Discussion**

Indicates the units for the frame's display time.

base

**Discussion**

Refers to the time base.

duration

**Discussion**

Specifies the duration for which the frame is to be displayed. This must be in the same units as specified by the `scale` field. It is 0 if the duration is unknown.

rate

**Discussion**

Indicates the time base's effective rate.

recordSize

**Discussion**

Total number of bytes in this structure.

frameNumber

**Discussion**

Number of frame; 0 if the frame number is not known.

flags

**Discussion**

Flag (see below) to indicate if `virtualStartTime` and `virtualDuration` are valid. See these constants:

`icmFrameTimeHasVirtualStartTimeAndDuration`

`icmFrameTimeHasDecodeTime`

virtualStartTime

**Discussion**

Conceptual start time.

virtualDuration

**Discussion**

Conceptual duration.

decodeTime

**Discussion**

Suggested decode time. Valid only if `icmFrameTimeHasDecodeTime` is set in the `flags` parameter.

**Programming Info**

C interface file: `ImageCompression.h`

## ICMProgressProcRecord

Specifies an image compression progress callback.

```
struct ICMProgressProcRecord {
    ICMProgressUPP    progressProc;
    long              progressRefCon;
};
```

**Fields**

`progressProc`

**Discussion**

Contains a pointer to your progress function.

`progressRefCon`

**Discussion**

Contains a reference constant for use by your progress function.

**Discussion**

During a compression operation, your compressor may occasionally call a function that the application provides in order to report your progress. This field contains a structure that identifies the progress function. If the `progressProc` field in this structure is set to `NIL`, the application has not supplied a progress function

**Related Functions**

[DrawPictureFile](#)  
[DrawTrimmedPicture](#)  
[DrawTrimmedPictureFile](#)  
[FCompressImage](#)  
[FCompressPicture](#)  
[FCompressPictureFile](#)  
[FDecompressImage](#)  
[GetCompressedPixMapInfo](#)  
[GraphicsExportGetProgressProc](#)  
[GraphicsExportSetProgressProc](#)  
[GraphicsImportGetProgressProc](#)  
[GraphicsImportSetProgressProc](#)  
[ImageCodecTrimImage](#)  
[MakeFilePreview](#)  
[MakeThumbnailFromPicture](#)  
[MakeThumbnailFromPictureFile](#)  
[MakeThumbnailFromPixMap](#)  
[PreviewMakePreview](#)



SetCompressedPixMapInfo  
SetSequenceProgressProc  
TrimImage

### Programming Info

C interface file: `ImageCompression.h`

## MatrixRecord

Contains a transformation matrix.

```
struct MatrixRecord {  
    Fixed    matrix[3][3];  
};
```

### Fields

`matrix`

### Discussion

A 3-by-3 array of matrix values.

### Related Functions

`GetMovieMatrix`  
`GetTrackMatrix`  
`GraphicsImportGetDefaultMatrix`  
`MediaSetMatrix`  
`TransformRgn`  
`TranslateMatrix`  
`VDSetPlayThruDestination`

### Programming Info

C interface file: `ImageCompression.h`

## MediaRecord

Undocumented

```
struct MediaRecord {  
    long    data[1];  
};
```

### Fields

`data`

### Discussion

*Undocumented*

### Programming Info

C interface file: `Movies.h`

## MovieRecord

Undocumented

```
struct MovieRecord {
    long    data[1];
};
```

**Fields**

data

**Discussion***Undocumented***Programming Info**C interface file: `Movies.h`**PixMap**

Contains information about the dimensions and contents of a pixel image, as well as its storage format, depth, resolution, and color usage.

```
struct PixMap {
    Ptr        baseAddr;
    short      rowBytes;
    Rect       bounds;
    short      pmVersion;
    short      packType;
    long       packSize;
    Fixed      hRes;
    Fixed      vRes;
    short      pixelType;
    short      pixelSize;
    short      cmpCount;
    short      cmpSize;
    OSType     pixelFormat;
    CTabHandle pmTable;
    void *     pmExt;
};
```

**Fields**

baseAddr

**Discussion**

For an onscreen pixel image, a pointer to the first byte of the image. For optimal performance, this should be a multiple of 4. The `baseAddr` field of the `PixMap` record for an offscreen graphics world contains a handle instead of a pointer. Your application should never directly access the `baseAddr` field of the `PixMap` record for an offscreen graphics world.

rowBytes

**Discussion**

The offset in bytes from one row of the image to the next. The value must be even, less than 0x4000, and for best performance it should be a multiple of 4. The high 2 bits of `rowBytes` are used as flags. If bit 15 = 1, the data structure pointed to is a `PixMap` structure; otherwise it is a `BitMap` structure.

`bounds`

**Discussion**

The boundary rectangle, which links the local coordinate system of a graphics port to QuickDraw's global coordinate system and defines the area of the bit image into which QuickDraw can draw. By default, the boundary rectangle is the entire main screen. Do not use the `value` of this field to determine the size of the screen; instead use the `value` of the `gdRect` field of the `GDevice` structure for the screen.

`pmVersion`

**Discussion**

The version number of Color QuickDraw that created this `PixelFormat` structure. The value of `pmVersion` is normally 0. If `pmVersion` is 4, Color QuickDraw treats the `PixelFormat` record's `baseAddr` field as 32-bit clean. All other flags are private. Most applications never need to set this field.

`packType`

**Discussion**

The packing algorithm used to compress image data. Color QuickDraw currently supports a `packType` of 0, which means no packing, and values of 1 to 4 for packing direct pixels.

`packSize`

**Discussion**

The size of the packed image in bytes. When the `packType` field contains the value 0, this field is always set to 0.

`hRes`

**Discussion**

The horizontal resolution of the pixel image in pixels per inch. By default, this value is 0x00480000 (for 72 pixels per inch).

`vRes`

**Discussion**

The vertical resolution of the pixel image in pixels per inch. By default, this value is 0x00480000 (for 72 pixels per inch).

`pixelType`

**Discussion**

The storage format for a pixel image. Indexed pixels are indicated by a value of 0. Direct pixels are specified by a value of `RGBDirect`, or 16. In the `PixelFormat` record of the `GDevice` structure for a direct device, this field is set to `RGBDirect` when the screen depth is set.

`pixelSize`

**Discussion**

The number of bits used to represent a pixel. Indexed pixels can have sizes of 1, 2, 4, and 8 bits; direct pixel sizes are 16 and 32 bits.

`cmpCount`

**Discussion**

The number of components used to represent a color for a pixel. With indexed pixels, each pixel is a single value representing an index in a color table, and therefore this field contains the value 1; the index is the single component. With direct pixels, each pixel contains three components (one integer each for the intensities of red, green, and blue) so this field contains the value 3.

`cmpSize`**Discussion**

The size in bits of each component for a pixel. Color QuickDraw expects that the sizes of all components are the same, and that the value of the `cmpCount` field multiplied by the value of the `cmpSize` field is less than or equal to the value in the `pixelSize` field.

For an indexed pixel value, which has only one component, the value of the `cmpSize` field is the same as the value of the `pixelSize` field; that is, 1, 2, 4, or 8. For direct pixels there are two additional possibilities. A 16-bit pixel, which has three components, has a `cmpSize` value of 5; this leaves an unused high-order bit, which Color QuickDraw sets to 0. A 32-bit pixel, which has three components (red, green, and blue), has a `cmpSize` value of 8; this leaves an unused high-order byte, which Color QuickDraw sets to 0.

If presented with a 32-bit image (for example, in the `CopyBits` procedure) Color QuickDraw passes whatever bits are there, and it does not set the high byte to 0. Generally, therefore, your application should clear the memory for the image to 0 before creating a 16-bit or 32-bit image.

`planeBytes`**Discussion**

The offset in bytes from one drawing plane to the next. This field is set to 0.

`pmTable`**Discussion**

A handle to a `ColorTable` structure for the colors in this pixel map.

`pmReserved`**Discussion**

Reserved. This field must be set to 0 for future compatibility.

`pixelFormat`**Discussion**

The way the pixels are arranged; see `Pixel Formats`.

`pmTable`**Discussion**

Color map for this structure.

`pmExt`**Discussion**

Handle to a `PixmapExtension` structure. Set to NIL if there is no extension.

**Discussion**

The pixel map for a window's color graphics port always consists of the pixel depth, color table, and boundary rectangle of the main screen, even if the window is created on or moved to an entirely different screen.

**Version Notes**

Earlier versions of this structure were different in the last three fields; see the C interface file for details.

**Programming Info**

C interface file: `Quickdraw.h`

**Point**

Defines the position of a point.

```
struct Point {
    short    v;
    short    h;
};
```

### Fields

v

### Discussion

The vertical coordinate of the point.

h

### Discussion

The horizontal coordinate of the point.

### Programming Info

C interface file: `MacTypes.h`

## QTEventRecord

Records a user event for QuickTime.

```
struct QTEventRecord {
    long        version;
    OSType      eventType;
    Point       where;
    long        flags;
};
```

### Fields

version

### Discussion

*Undocumented*

eventType

### Discussion

*Undocumented*

where

### Discussion

The location of the cursor at the time the event was posted.

flags

### Discussion

*Undocumented*

### Discussion

This structure is used by the `kActionSendQTEventToSprite` action.

### Related Functions

`ActionsProc`

`CallComponentExecuteWiredAction`

`MediaGetActionsForQTEvent`

`SpriteMediaGetSpriteActionsForQTEvent`

### Programming Info

C interface file: `Movies.h`

## Rect

Defines the size and location of a QuickDraw rectangle.

```
struct Rect {
    short    top;
    short    left;
    short    bottom;
    short    right;
};
```

### Fields

`top`

### Discussion

The vertical coordinate of the upper-left point of the rectangle.

`left`

### Discussion

The horizontal coordinate of the upper-left point of the rectangle.

`bottom`

### Discussion

The vertical coordinate of the lower-right point of the rectangle.

`right`

### Discussion

The horizontal coordinate of the lower-right point of the rectangle.

### Programming Info

C interface file: `Quickdraw.h`

## RGBColor

Defines a color in the red-green-blue system.

```
struct RGBColor {
    unsigned short    red;
    unsigned short    green;
    unsigned short    blue;
};
```

### Fields

`red`

### Discussion

The magnitude of the red component

`green`

### Discussion

The magnitude of the green component

blue

### Discussion

The magnitude of the blue component

### Related Functions

GraphicsImportGetGraphicsMode  
MediaGetGraphicsMode  
SGSetTextForeColor  
TextMediaAddTextSample  
TextMediaHiliteTextSample  
VDGetKeyColorRange

### Programming Info

C interface file: Quickdraw.h

## TimeBaseRecord

Contains a time base.

```
struct TimeBaseRecord {  
    long    data[1];  
};
```

### Fields

data

### Discussion

Array of data that constitutes a time base.

### Programming Info

C interface file: Movies.h

## TimeRecord

Contains a time value with its scale and time base.

```
struct TimeRecord {  
    CompTimeValue    value;  
    TimeScale        scale;  
    TimeBase         base;  
};
```

### Fields

value

### Discussion

Contains the time value. The time value defines either a duration or an absolute time by specifying the corresponding number of units of time. For durations, this is the number of time units in the period. For an absolute time, this is the number of time units since the beginning of the time coordinate system. The unit for this value is defined by the `scale` field. The time value is expressed as a 64-bit integer quantity. This 64-bit quantity consists of two 32-bit integers and is defined by the `Int64` data type.

scale

#### Discussion

Contains the `time` scale. This field specifies the number of units of time that pass each second. If you specify a value of 0, the time base uses its natural time scale.

base

#### Discussion

Contains a reference to the time base. You obtain a time base by calling `GetMovieTimeBase` or `NewTimeBase`. If the time structure defines a duration, set this field to `NIL`. Otherwise, this field must refer to a valid time base.

#### Related Functions

`AddTime`  
`ClockGetTime`  
`GetMovieTime`  
`GetTimeBaseStartTime`  
`GetTimeBaseStatus`  
`GetTimeBaseStopTime`  
`GetTimeBaseTime`  
`SetTimeBaseZero`  
`SGGrabCompressComplete`  
`SubtractTime`  
`VDCompressDone`  
`VDGetTimeCode`

#### Programming Info

C interface file: `Movies.h`

## ToneDescription

Provides the information needed to produce a specific musical sound.

```
struct ToneDescription {
    BigEndianOSType    synthesizerType;
    Str31              synthesizerName;
    Str31              instrumentName;
    BigEndianLong      instrumentNumber;
    BigEndianLong      gmNumber;
};
```

#### Fields

`synthesizerType`

#### Discussion

A synthesizer type constant (see below). A value of 0 specifies that any type of synthesizer is acceptable. See these constants:

`kSoftSynthComponentSubType`  
`kGMSynthComponentSubType`

`synthesizerName`

#### Discussion

The name of the instrument to use.



`instrumentName`

#### Discussion

The name of the instrument to use.

`instrumentNumber`

#### Discussion

The instrument number of the instrument to use. This value, which must be in the range 1-262143, can specify General MIDI and GS instruments as well as other instruments. The instrument specified by this field is used if it is available; if not, the instrument specified by the `gmNumber` field is used. If neither of the instruments specified by the `instrumentNumber` or `gmNumber` fields is available, the instrument specified by the `instrumentName` field is used. Finally, if none of these fields specifies an instrument that is available, no tone is played.

`gmNumber`

#### Discussion

The instrument number of a General MIDI or GS instrument to use if the instrument specified by the `instrumentNumber` field is not available. This value, which must be in the range 1-16383, can specify only General MIDI and GS instruments. The instrument specified by the `instrumentNumber` field is used if it is available; if not, the instrument specified by the `gmNumber` field is used. If neither of the instruments specified by the `instrumentNumber` or `gmNumber` fields is available, the instrument specified by the `instrumentName` field is used. Finally, if none of these fields specifies an instrument that is available, no tone is played.

#### Discussion

The tune header in the QuickTime Music Architecture has a `ToneDescription` structure for each instrument used. These structures are also used in the tone description atoms of atomic instruments.

#### Related Functions

`MusicFindTone`  
`NAFindNoteChannelTone`  
`NAPickInstrument`  
`NAStuffToneDescription`  
`SGGetInstrument`  
`SGSetInstrument`

#### Programming Info

C interface file: `QuickTimeMusic.h`

## TrackRecord

Contains a track.

```
struct TrackRecord {
    long    data[1];
};
```

#### Fields

`data`

#### Discussion

An array of track data.

#### Programming Info

C interface file: `Movies.h`

**UserDataRecord**

Contains user data.

```
struct UserDataRecord {
    long    data[1];
};
```

**Fields**

data

**Discussion**

An array of user data.

**Discussion**

Use `NewUserData` to create this record and `DisposeUserData` to dispose of it.

**Related Functions**

`NewUserData`

`DisposeUserData`

**Programming Info**

C interface file: `Movies.h`

**wide**

Stores a signed 64-bit value as a signed 32-bit integer and an unsigned 32-bit integer.

```
struct wide {          // big-endian version
    SInt32    hi;
    UInt32    lo;
};
struct wide {          // little-endian version
    UInt32    lo;
    SInt32    hi;
};
```

**Fields**

hi

**Discussion**

The signed high-order 32-bit integer.

lo

**Discussion**

The unsigned low-order 32-bit integer.

**Programming Info**

C interface file: `Endian.h`

**ActionsUPP**

`Abst_ActionsUPP`

```
typedef STACK_UPP_TYPE(ActionsProcPtr) ActionsUPP;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

**AliasHandle**

Abst\_AliasHandle

```
typedef AliasPtr * AliasHandle;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Aliases.h

**AliasPtr**

Abst\_AliasPtr

```
typedef AliasRecord * AliasPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Aliases.h

**ByteCount**

Abst\_ByteCount

```
typedef UInt32 ByteCount;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

IOHIDDescriptorParser.h

**CGrafPtr**

Abst\_CGrafPtr

```
typedef CGrafPort * CGrafPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

QuickdrawTypes.h

**CodecQ**

Abst\_CodecQ

```
typedef unsigned long CodecQ;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

ImageCompression.h

**CodecType**

Abst\_CodecType

```
typedef OSType CodecType;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

ImageCompression.h

**ComponentInstance**

Abst\_ComponentInstance

```
typedef ComponentInstanceRecord * ComponentInstance;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Components.h

**ComponentResult**

Abst\_ComponentResult

```
typedef long ComponentResult;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Components.h

## CompressorComponent

Abst\_CompressorComponent

```
typedef Component CompressorComponent;
```

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

ImageCompression.h

## ConstStr255Param

Abst\_ConstStr255Param

```
typedef const unsigned char * ConstStr255Param;
```

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

MacTypes.h

## CTabHandle

Abst\_CTabHandle

```
typedef CTabPtr * CTabHandle;
```

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

QuickdrawTypes.h

## CTabPtr

Abst\_CTabPtr

```
typedef ColorTable * CTabPtr;
```

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

QuickdrawTypes.h

## DataHandler

Abst\_DataHandler

```
typedef ComponentInstance DataHandler;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

## DialogPtr

Abst\_DialogPtr

```
typedef WindowPtr DialogPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

QuickdrawTypes.h

## DialogRef

Abst\_DialogRef

```
typedef DialogPtr DialogRef;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Dialogs.h

## DoMCActionUPP

Abst\_DoMCActionUPP

```
typedef STACK_UPP_TYPE(DoMCActionProcPtr) DoMCActionUPP;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

## GDHandle

Abst\_GDHandle

```
typedef GDPtr * GDHandle;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

QuickdrawTypes.h

**GDPtr**

Abst\_GDPtr

```
typedef GDevice * GDPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

QuickdrawTypes.h

**GWorldFlags**

Abst\_GWorldFlags

```
typedef unsigned long GWorldFlags;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

QDOffscreen.h

**GWorldPtr**

Abst\_GWorldPtr

```
typedef CGrafPtr GWorldPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

QDOffscreen.h

**ICMAalignmentProcRecordPtr**

Abst\_ICMAalignmentProcRecordPtr

```
typedef ICMAalignmentProcRecord * ICMAalignmentProcRecordPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

ImageCompression.h

## **ICMCompletionProcRecordPtr**

Abst\_ICMCompletionProcRecordPtr

```
typedef ICMCompletionProcRecord * ICMCompletionProcRecordPtr;
```

### **Availability**

Available in Mac OS X v10.0 and later.

### **Declared In**

ImageCompression.h

## **ICMConvertDataFormatUPP**

Abst\_ICMConvertDataFormatUPP

```
typedef STACK_UPP_TYPE(ICMConvertDataFormatProcPtr) ICMConvertDataFormatUPP;
```

### **Availability**

Available in Mac OS X v10.0 and later.

### **Declared In**

ImageCompression.h

## **ICMDataProcRecordPtr**

Abst\_ICMDataProcRecordPtr

```
typedef ICMDataProcRecord * ICMDataProcRecordPtr;
```

### **Availability**

Available in Mac OS X v10.0 and later.

### **Declared In**

ImageCompression.h

## **ICMFlushProcRecordPtr**

Abst\_ICMFlushProcRecordPtr

```
typedef ICMFlushProcRecord * ICMFlushProcRecordPtr;
```

### **Availability**

Available in Mac OS X v10.0 and later.

### **Declared In**

ImageCompression.h

## **ICMMemoryDisposedUPP**

Abst\_ICMMemoryDisposedUPP



```
typedef STACK_UPP_TYPE(ICMemoryDisposedProcPtr) ICMemoryDisposedUPP;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

ImageCompression.h

## ICMProgressProcRecordPtr

Abst\_ICMProgressProcRecordPtr

```
typedef ICMProgressProcRecord * ICMProgressProcRecordPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

ImageCompression.h

## ImageDescriptionHandle

Abst\_ImageDescriptionHandle

```
typedef ImageDescriptionPtr * ImageDescriptionHandle;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

ImageCompression.h

## ImageDescriptionPtr

Abst\_ImageDescriptionPtr

```
typedef ImageDescription * ImageDescriptionPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

ImageCompression.h

## ImageSequence

Abst\_ImageSequence

```
typedef long ImageSequence;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

ImageCompression.h

**ItemCount**

Abst\_ItemCount

```
typedef UInt32 ItemCount;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

IOMacOSTypes.h

**MatrixRecordPtr**

Abst\_MatrixRecordPtr

```
typedef MatrixRecord * MatrixRecordPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

ImageCompression.h

**Media**

Abst\_Media

```
typedef MediaRecord * Media;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

**MediaHandler**

Abst\_MediaHandler

```
typedef ComponentInstance MediaHandler;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

## MenuHandle

Abst\_MenuHandle

```
typedef MenuPtr * MenuHandle;
```

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

Menus.h

## MenuRef

Abst\_MenuRef

```
typedef MenuHandle MenuRef;
```

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

Menus.h

## ModalFilterUPP

Abst\_ModalFilterUPP

```
typedef STACK_UPP_TYPE(ModalFilterProcPtr) ModalFilterUPP;
```

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

Dialogs.h

## Movie

Abst\_Movie

```
typedef MovieRecord * Movie;
```

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

Movies.h

## MovieController

Abst\_MovieController

```
typedef ComponentInstance MovieController;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

## **MovieDrawingCompleteUPP**

Abst\_MovieDrawingCompleteUPP

```
typedef STACK_UPP_TYPE(MovieDrawingCompleteProcPtr) MovieDrawingCompleteUPP;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

## **MoviePrePrerollCompleteUPP**

Abst\_MoviePrePrerollCompleteUPP

```
typedef STACK_UPP_TYPE(MoviePrePrerollCompleteProcPtr) MoviePrePrerollCompleteUPP;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

## **MoviePreviewCallOutUPP**

Abst\_MoviePreviewCallOutUPP

```
typedef STACK_UPP_TYPE(MoviePreviewCallOutProcPtr) MoviePreviewCallOutUPP;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

## **MovieProgressUPP**

Abst\_MovieProgressUPP

```
typedef STACK_UPP_TYPE(MovieProgressProcPtr) MovieProgressUPP;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

**MoviesErrorUPP**

Abst\_MoviesErrorUPP

```
typedef STACK_UPP_TYPE(MoviesErrorProcPtr) MoviesErrorUPP;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

**OSErr**

Abst\_OSErr

```
typedef SInt16 OSErr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

IOMacOSTypes.h

**OSStatus**

Abst\_OSStatus

```
typedef SInt32 OSStatus;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

OSTypes.h

**PicHandle**

Abst\_PicHandle

```
typedef PicPtr * PicHandle;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

QuickdrawTypes.h

## **PicPtr**

Abst\_PicPtr

```
typedef Picture * PicPtr;
```

### **Availability**

Available in Mac OS X v10.0 and later.

### **Declared In**

QuickdrawTypes.h

## **PixMapHandle**

Abst\_PixMapHandle

```
typedef PixMapPtr * PixMapHandle;
```

### **Availability**

Available in Mac OS X v10.0 and later.

### **Declared In**

QuickdrawTypes.h

## **PixMapPtr**

Abst\_PixMapPtr

```
typedef PixMap * PixMapPtr;
```

### **Availability**

Available in Mac OS X v10.0 and later.

### **Declared In**

QuickdrawTypes.h

## **QTAtom**

Abst\_QTAtom

```
typedef long QTAtom;
```

### **Availability**

Available in Mac OS X v10.0 and later.

### **Declared In**

Movies.h

## **QTAtomContainer**

Abst\_QTAtomContainer

```
typedef Handle QTAtomContainer;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

## QTAtomID

Abst\_QTAtomID

```
typedef long QTAtomID;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

## QTCallback

Abst\_QTCallback

```
typedef CallbackRecord * QTCallback;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

## QTCallbackUPP

Abst\_QTCallbackUPP

```
typedef STACK_UPP_TYPE(QTCallbackProcPtr) QTCallbackUPP;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

## QTEventRecordPtr

Abst\_QTEventRecordPtr

```
typedef QTEventRecord * QTEventRecordPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

**QTNextTaskNeededSoonerCallbackUPP**

Abst\_QTNextTaskNeededSoonerCallbackUPP

```
typedef STACK_UPP_TYPE(QTNextTaskNeededSoonerCallbackProcPtr)  
QTNextTaskNeededSoonerCallbackUPP;
```

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

Movies.h

**QTParameterDialog**

Abst\_QTParameterDialog

```
typedef long QTParameterDialog;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

**QTParameterDialogOptions**

Abst\_QTParameterDialogOptions

```
typedef long QTParameterDialogOptions;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

**RgnHandle**

Abst\_RgnHandle

```
typedef RgnPtr * RgnHandle;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

QuickdrawTypes.h



## **RgnPtr**

Abst\_RgnPtr

```
typedef MacRegion * RgnPtr;
```

### **Availability**

Available in Mac OS X v10.0 and later.

### **Declared In**

QuickdrawTypes.h

## **SampleDescriptionHandle**

Abst\_SampleDescriptionHandle

```
typedef SampleDescriptionPtr * SampleDescriptionHandle;
```

### **Availability**

Available in Mac OS X v10.0 and later.

### **Declared In**

Movies.h

## **SampleDescriptionPtr**

Abst\_SampleDescriptionPtr

```
typedef SampleDescription * SampleDescriptionPtr;
```

### **Availability**

Available in Mac OS X v10.0 and later.

### **Declared In**

Movies.h

## **ScriptCode**

Abst\_ScriptCode

```
typedef SInt16 ScriptCode;
```

### **Availability**

Available in Mac OS X v10.0 and later.

### **Declared In**

MacTypes.h

## **Size**

Abst\_Size

```
typedef long Size;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

MacTypes.h

## SoundDescriptionHandle

Abst\_SoundDescriptionHandle

```
typedef SoundDescriptionPtr * SoundDescriptionHandle;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

## SoundDescriptionPtr

Abst\_SoundDescriptionPtr

```
typedef SoundDescription * SoundDescriptionPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

## Str255

Abst\_Str255

```
typedef unsigned char Str255;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

MacTypes.h

## StringPtr

Abst\_StringPtr

```
typedef unsigned char * StringPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

MacTypes.h

**TextMediaUPP**

Abst\_TextMediaUPP

```
typedef STACK_UPP_TYPE(TextMediaProcPtr) TextMediaUPP;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

Movies.h

**TimeBase**

Abst\_TimeBase

```
typedef TimeBaseRecord * TimeBase;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

MacTypes.h

**TimeScale**

Abst\_TimeScale

```
typedef long TimeScale;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

MacTypes.h

**TimeValue**

Abst\_TimeValue

```
typedef long TimeValue;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

MacTypes.h

## TimeValue64

Abst\_TimeValue64

```
typedef SInt64 TimeValue64;
```

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

MacTypes.h

## Track

Abst\_Track

```
typedef TrackRecord * Track;
```

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

Movies.h

## TrackTransferUPP

Abst\_TrackTransferUPP

```
typedef STACK_UPP_TYPE(TrackTransferProcPtr) TrackTransferUPP;
```

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

Movies.h

## UserData

Abst\_UserData

```
typedef UserDataRecord * UserData;
```

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

Movies.h

## VdigIntUPP

Abst\_VdigIntUPP

```
typedef STACK_UPP_TYPE(VdigIntProcPtr) VdigIntUPP;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

QuickTimeComponents.h

## WindowPtr

Abst\_WindowPtr

```
typedef GrafPtr WindowPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

QuickdrawTypes.h

## WindowRef

Abst\_WindowRef

```
typedef WindowPtr WindowRef;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

QuickdrawTypes.h



# Document Revision History

---

This table describes the changes to *QuickTime Data Types Reference*.

| Date       | Notes   |
|------------|---|
| 2006-05-23 | New document, based on previously published material, that covers data types common to multiple QuickTime frameworks. |

## REVISION HISTORY

Document Revision History



# Index

---

## A

---

ActionsUPP **data type** [34](#)  
AliasHandle **data type** [35](#)  
AliasPtr **data type** [35](#)

## B

---

ByteCount **data type** [35](#)

## C

---

CallbackRecord **structure** [7](#)  
CGrafPort **structure** [8](#)  
CGrafPtr **data type** [35](#)  
CodecInfo **structure** [12](#)  
CodecQ **data type** [36](#)  
CodecType **data type** [36](#)  
ComponentInstance **data type** [36](#)  
ComponentInstanceRecord **structure** [17](#)  
ComponentResult **data type** [36](#)  
CompressorComponent **data type** [37](#)  
ConstStr255Param **data type** [37](#)  
CTabHandle **data type** [37](#)  
CTabPtr **data type** [37](#)

## D

---

DataHandler **data type** [37](#)  
DialogPtr **data type** [38](#)  
DialogRef **data type** [38](#)  
DoMCActionUPP **data type** [38](#)

## E

---

EventRecord **structure** [17](#)

## F

---

FixedPoint **structure** [18](#)  
FSSpec **structure** [19](#)

## G

---

GDHandle **data type** [38](#)  
GDPtr **data type** [39](#)  
GWorldFlags **data type** [39](#)  
GWorldPtr **data type** [39](#)

## I

---

ICMAlignmentProcRecord **structure** [19](#)  
ICMAlignmentProcRecordPtr **data type** [39](#)  
ICMCompletionProcRecord **structure** [20](#)  
ICMCompletionProcRecordPtr **data type** [40](#)  
ICMConvertDataFormatUPP **data type** [40](#)  
ICMDataProcRecord **structure** [21](#)  
ICMDataProcRecordPtr **data type** [40](#)  
ICMFlushProcRecord **structure** [22](#)  
ICMFlushProcRecordPtr **data type** [40](#)  
ICMFrameTimeRecord **structure** [22](#)  
ICMMemoryDisposedUPP **data type** [40](#)  
ICMProgressProcRecord **structure** [24](#)  
ICMProgressProcRecordPtr **data type** [41](#)  
ImageDescriptionHandle **data type** [41](#)  
ImageDescriptionPtr **data type** [41](#)  
ImageSequence **data type** [41](#)  
ItemCount **data type** [42](#)

## M

---

MatrixRecord **structure** [25](#)  
 MatrixRecordPtr **data type** [42](#)  
 Media **data type** [42](#)  
 MediaHandler **data type** [42](#)  
 MediaRecord **structure** [25](#)  
 MenuHandle **data type** [43](#)  
 MenuRef **data type** [43](#)  
 ModalFilterUPP **data type** [43](#)  
 Movie **data type** [43](#)  
 MovieController **data type** [43](#)  
 MovieDrawingCompleteUPP **data type** [44](#)  
 MoviePrePrerollCompleteUPP **data type** [44](#)  
 MoviePreviewCallOutUPP **data type** [44](#)  
 MovieProgressUPP **data type** [44](#)  
 MovieRecord **structure** [25](#)  
 MoviesErrorUPP **data type** [45](#)

## O

---

OSErr **data type** [45](#)  
 OSStatus **data type** [45](#)

## P

---

PicHandle **data type** [45](#)  
 PicPtr **data type** [46](#)  
 PixMap **structure** [26](#)  
 PixMapHandle **data type** [46](#)  
 PixMapPtr **data type** [46](#)  
 Point **structure** [28](#)

## Q

---

QTAtom **data type** [46](#)  
 QTAtomContainer **data type** [46](#)  
 QTAtomID **data type** [47](#)  
 QTCallback **data type** [47](#)  
 QTCallbackUPP **data type** [47](#)  
 QTEventRecord **structure** [29](#)  
 QTEventRecordPtr **data type** [47](#)  
 QTNextTaskNeededSoonerCallbackUPP **data type** [48](#)  
 QTParameterDialog **data type** [48](#)  
 QTParameterDialogOptions **data type** [48](#)

## R

---

Rect **structure** [30](#)  
 RGBColor **structure** [30](#)  
 RgnHandle **data type** [48](#)  
 RgnPtr **data type** [49](#)

## S

---

SampleDescriptionHandle **data type** [49](#)  
 SampleDescriptionPtr **data type** [49](#)  
 ScriptCode **data type** [49](#)  
 Size **data type** [49](#)  
 SoundDescriptionHandle **data type** [50](#)  
 SoundDescriptionPtr **data type** [50](#)  
 Str255 **data type** [50](#)  
 StringPtr **data type** [50](#)

## T

---

TextMediaUPP **data type** [51](#)  
 TimeBase **data type** [51](#)  
 TimeBaseRecord **structure** [31](#)  
 TimeRecord **structure** [31](#)  
 TimeScale **data type** [51](#)  
 TimeValue **data type** [51](#)  
 TimeValue64 **data type** [52](#)  
 ToneDescription **structure** [32](#)  
 Track **data type** [52](#)  
 TrackRecord **structure** [33](#)  
 TrackTransferUPP **data type** [52](#)

## U

---

UserData **data type** [52](#)  
 UserDataRecord **structure** [34](#)

## V

---

VdigIntUPP **data type** [52](#)

## W

---

wide **structure** [34](#)  
 WindowPtr **data type** [53](#)

WindowRef data type [53](#)