Carbon Window Manager API Preliminary Documentation

For CarbonLib 1.0

Preliminary Draft

Technical Publications

In the following sections, this document discusses the changes to the Window Manager that are part of the Carbon library which is included with Mac OS 9.0:

- "Window Classes and Layering" (page 4)
- "Changing Window Attributes" (page 6)
- "Window Scrolling" (page 7)
- "Custom Window Definitions" (page 10)
- "Erasing the Window's Content Region" (page 12)

IMPORTANT

This is a preliminary document. Although it has been reviewed for technical accuracy, it is not final. Apple Computer, Inc. is supplying this information to help you plan for the adoption of the technologies and programming interfaces described herein. This information is subject to change, and software implemented according to this document should be tested with final operating system

You can check <http://developer.apple.com/techpubs/macos8/SiteInfo/whatsnew.html> for information about updates to this and other developer documents. To receive notification of documentation updates, you can sign up for ADC's free Online Program and receive their weekly Apple Developer Connection News e-mail newsletter. (See <http://developer.apple.com/membership/index.html> for more details about the Online Program.) ▲

Window Classes and Layering

FindWindowOfClass

Obtains the window at the specified point onscreen.

```
pascal OSStatus FindWindowOfClass( const Point * inWhere, WindowClass
                        inWindowClass, WindowRef * outWindow, SInt16 *
                        outWindowPart )
```

`inWhere`	Global point at which to search for a window.
`inWindowClass`	If the window found at the specified point is not a window of this specific class, `errWindowNotFound` is returned. If the constant `kAllWindowClasses` is passed, returns any window found at `inWhere`.
`outWindow`	If a window is found, it is returned here. Otherwise, `NULL` is returned. This parameter can be `NULL`.
`outWindowPart`	If a window is found, the part code of the window part under the mouse is returned here. If no window is found, `inDesk` is returned. This parameter can be `NULL`.
function result	An error code. If a window is found, `noErr` is returned. Otherwise, `errWindowNotFound` is returned.

DISCUSSION

A version of the `FindWindow` function which allows the developer to limit the search to windows of one particular class. If a window is found at the specified point, but is not of the specified `WindowClass`, `errWindowNotFound` is returned and the value of `outWindow` is set to `NULL`.

VERSION NOTES

Available from CarbonLib 1.0 forward, running on Mac OS 8.1 and later.

GetFrontWindowOfClass

Return the front window of the specified class

```
pascal WindowRef GetFrontWindowOfClass( WindowClass inWindowClass,
                        Boolean inMustBeVisible );
```

`inWindowClass`	A value of type `WindowClass`. Pass a valid window class to get the frontmost window of belonging to that class. If `kAllWindowClasses` is passed, the frontmost window in the window list is returned.
`inMustBeVisible`	A value of type `Boolean`. Pass `true` to retrieve only visible windows; pass `false` to retrieve all windows.

DISCUSSION

A more explicit version of the `FrontWindow` and `FrontNonFloatingWindow` functions.

VERSION NOTES

Available from CarbonLib 1.0 forward, running on Mac OS 8.1 and later.

GetNextWindowOfClass

Returns the first window behind the specified window which is of the specified `WindowClass`.

```
pascal WindowRef GetNextWindowOfClass( WindowRef inWindow, WindowClass
                        inWindowClass, Boolean inMustBeVisible );
```

`inWindow`	A value of type `WindowRef`. Pass a valid window reference.
`inWindowClass`	A value of type `WindowClass`. Pass a valid window class to get the next window of belonging to that class. If `kAllWindowClasses` is passed, the window directly behind the input window is returned. If there are no windows of the specified class behind the input window, `NULL` is returned.

`inMustBeVisible`
A value of type `Boolean`. Pass `true` to retrieve only visible windows; pass `false` to retrieve all windows.

DISCUSSION

A more explicit version of the function `GetNextWindow`.

VERSION NOTES

Available from CarbonLib 1.0 forward, running on Mac OS 8.1 and later.

Changing Window Attributes

ChangeWindowAttributes

Change any attributes of a window.

```
pascal OSStatus ChangeWindowAttributes( WindowRef window,
                    WindowAttributes attributesToSet, WindowAttributes
                    attributesToClear );
```

`window` A value of type `WindowRef`. Pass a valid window reference.

`attributesToSet`
A bit mask of type `WindowAttributes`. The specified attributes will be added to the window. Specify `kNoWindowAttributes` if you do not wish to add attributes.

`attributesToClear`
A bit mask of type `WindowAttributes`. The specified attributes will be removed from the window. Specify `kNoWindowAttributes` if you do not wish to remove attributes.

function result If the window's class cannot accept the specified attributes, `errUnsupportedWindowAttributesForClass` is returned. If the window is not a valid window ref, `errInvalidWindowRef` is returned. Otherwise `noErr`.

DISCUSSION

If the changed attributes affect the visible window frame, the window's regions will be recalculated, and the window will be redrawn onscreen.

SPECIAL CONSIDERATIONS

With the function `ChangeWindowAttributes` comes a change: `GetWindowAttributes` and `GetWindowClass` will return synthesized, accurate attributes and class when given a window created with 1984 window creation APIs (`New[C]Window`). In 8.5 and 8.6, `GetWindowAttributes` returns `paramErr` unless the window was created using `CreateNewWindow`. Also, the class will be accurate whether or not the client has called `InitFloatingWindows`.

VERSION NOTES

Available from CarbonLib 1.0 forward, running on Mac OS 8.1 and later.

Window Scrolling

ScrollWindowRect

Scroll any area of a window

```
pascal OSStatus ScrollWindowRect( WindowRef inWindow, const Rect *
                        inScrollRect, SInt16 inHPixels, SInt16 inVPixels,
                        ScrollWindowOptions inOptions, RgnHandle
                        outExposedRgn );
```

`inWindow`	A value of type `WindowRef`. Pass a valid window reference.
`inScrollRect`	The rectangle to scroll, in coordinates local to `inWindow`.
`inHPixels`	Pixels to scroll horizontally.
`inVPixels`	Pixels to scroll vertically.

`inOptions`	One or more options may be specified: `kScrollWindowInvalidate`: if set, the exposed area will be added to the window's update region. `kScrollWindowEraseToPortBackground`: if set, the exposed area will be redrawn with the grafport's background image (color or pattern).
`outExposedRgn`	Valid `RgnHandle` for the area newly revealed by the scroll (can be `NULL`; if `NULL`, the exposed region is added to the window's update region, regardless of the state of the `kScrollWindowInvalidate` option. This prevents updates from being lost in multiple-monitor situations where the Window Manager can't copy the entire region due to differing color tables).
function result	Memory errors, parameter errors. Otherwise `noErr`.

DISCUSSION

Scrolls pixels that are inside the specified region of the input window. No other pixels or the bits they represent are affected. The pixels are shifted a distance of `inHPixels` horizontally and `inVPixels` vertically. The positive directions are to the right and down. The pixels that are shifted out of the specified window are not displayed, and the bits they represent are not saved. The exposed empty area created by the scrolling is returned in the update region parameter and optionally added to the window's update region.

VERSION NOTES

Available from CarbonLib 1.0 forward, running on Mac OS 8.1 and later.

ScrollWindowRegion

Scroll any area of a window

```
pascal OSStatus ScrollWindowRegion( WindowRef inWindow, RgnHandle
                    inScrollRgn, SInt16 inHPixels, SInt16 inVPixels,
                    ScrollWindowOptions inOptions, RgnHandle
                    outExposedRgn);
```

`inWindow`	A value of type `WindowRef`. Pass a valid window reference.

`inScrollRgn`	The region to scroll, in coordinates local to `inWindow`.
`inHPixels`	Pixels to scroll horizontally.
`inVPixels`	Pixels to scroll vertically.
`inOptions`	One or more options may be specified: `kScrollWindowInvalidate`: if set, the exposed area will be added to the window's update region. `kScrollWindowEraseToPortBackground`: if set, the exposed area will be redrawn with the grafport's background image (color or pattern).
`outExposedRgn`	Valid `RgnHandle` for the area newly revealed by the scroll (can be `NULL`; if `NULL`, the exposed region is added to the window's update region, regardless of the state of the `kScrollWindowInvalidate` option. This prevents updates from being lost in multiple-monitor situations where the Window Manager can't copy the entire region due to differing color tables).
function result	Memory errors, parameter errors. Otherwise `noErr`.

DISCUSSION

Scrolls pixels that are inside the specified region of the input window. No other pixels or the bits they represent are affected. The pixels are shifted a distance of `inHPixels` horizontally and `inVPixels` vertically. The positive directions are to the right and down. The pixels that are shifted out of the specified window are not displayed, and the bits they represent are not saved. The exposed empty area created by the scrolling is returned in the update region parameter and optionally added to the window's update region.

VERSION NOTES

Available from CarbonLib 1.0 forward, running on Mac OS 8.1 and later.

Custom Window Definitions

CreateCustomWindow

Create a window with a custom structure region.

```
pascal OSStatus CreateCustomWindow( const WindowDefSpec *spec,
                    WindowClass windowClass, WindowAttributes
                    attributes, const Rect *contentBounds, WindowRef
                    *outWindow );
```

`spec`	Pointer to a structure of type `WindowDefSpec`. Specifies the window definition. For `kWindowDefProc`, the "defProc" field should contain a pointer to your application's window definition procedure.
`windowClass`	A value of type `WindowClass`. For custom windows, this determines the z-order layering of the window.
`attributes`	A bit mask of type `WindowAttributes`.
`contentBounds`	Pointer to a rectangle in global coordinates. Pass the initial rectangle of the window's content region (which will determine the size of the associated QuickDraw port's bounds).
`outWindow`	A pointer to a `WindowRef`. On return, this contains the newly-created window (if the function result is `noErr`).
function result	Memory errors, parameter errors. Otherwise `noErr`.

DISCUSSION

Create a window with a custom defproc. The only supported `WindowDef` type in Carbon 1.0 is type zero, `kWindowDefProc`, which is a classic 1984-style defproc.

```
typedef UInt32 WindowDefType;
typedef struct
{
    WindowDefType defType;
    union
    {
        WindowDefUPP defProc; // 1984-style original defproc
    } u;
} WindowDefSpec;
```

There are two major changes in the defproc calling model:

- The window definition should always draw into the current port, which is guaranteed to be a color port and may in fact be an offscreen GWorld.
- Because the `WindowRef` is opaque in Carbon, the window's regions are not directly accessible, so the `wCalc` message is not usable. When the window manager wants to recalculate the window's regions, it calls the defproc with `kWindowMsgGetRegions`, seperately requesting the content and the structure region. See *Mac OS 8 Window Manager Reference* for more info on `kWindowMsgGetRegions`.

VERSION NOTES

Available from CarbonLib 1.0 forward, running on Mac OS 8.1 and later.

ReshapeCustomWindow

If the shape of a custom window needs to change dynamically, outside of the context of normal Window Manager operations, you must use `ReshapeCustomWindow` to notify the Window Manager so that it can recalculate the window regions and update the screen.

```
pascal OSStatus ReshapeCustomWindow( WindowRef window );
```

`window`	A value of type `WindowRef`. Pass a valid window reference.
function result	Memory errors, `errInvalidWindowRef`, `noErr`.

DISCUSSION

`ReshapeCustomWindow` recalculates the shape of a custom window (by calling the WDEF with `kWindowMsgGetRegions` for the structure and content regions) and updates the screen with the window shape (redrawing the window and any windows behind it).

Prior to Carbon, it was possible to call `SaveOld`, modify the window structure and content regions, then call `DrawNew` to dynamically change the window shape. In Carbon, the `WindowRef` is opaque, so the window regions cannot be directly modified by the window definition. `ReshapeCustomWindow` replaces `SaveOld` and `DrawNew`.

VERSION NOTES

Available from CarbonLib 1.0 forward, running on Mac OS 8.1 and later.

Erasing the Window's Content Region

MyWindowPaintProc

Application-defined routine which paints the content region of a window.

```
typedef pascal OSStatus (*WindowPaintProcPtr)( GDHandle device, WindowRef
                     window, GrafPtr qdContext, RgnHandle inYouPaintRgn,
                     RgnHandle outSystemPaintRgn, void *refCon );
```

`device`	The current GDevice. `inYouPaintRgn` is contained within the bounds of this device.
`window`	A value of type `WindowRef`. Pass a valid window reference.
`context`	A QuickDraw graphics context; draw into this GrafPort, not the one associated with the window. `inYouPaintRgn` is defined relative to this GrafPort. Note that this might be an offscreen GWorld.
`regionCode`	Either `kWindowContentRegion` or `kWindowStructureRgn`
`inYouPaintRgn`	The region in need of painting. Treat as const. This region is clipped to the intersection of the current GDevice and the `clobberedRgn` passed to `PaintBehind`.

`outSystemPaintRgn`	Initially empty; when the PaintProc returns, the Window Manager will erase this region using the system window content paint proc.
`refCon`	A 32-bit reference value for your use.

DISCUSSION

Each window in the system contains a reference to a content paint proc. This proc is called to erase the window's content region during `PaintBehind` or `PaintOne` operations. The client application can override the system paint proc by calling `SetWindowContentPaintProc`. A window may only have one paintproc installed at any one time, and the paint proc cannot be retrieved by the client application.

If the content region PaintProc returns any value other than `noErr`, `outSystemPaintRgn` is ignored and the entire area of `inYouPaintRgn` is painted using the system paint proc.

When a previously obscured portion of a window is exposed, the window manager will iterate over active displays and call the window's content paint proc once for each device intersecting the region.

InstallWindowContentPaintProc

Associates your paint proc with the given window.

```
pascal OSStatus InstallWindowContentPaintProc( WindowRef window,
                    WindowPaintProcPtr paintProc, WindowPaintProcOptions
                    options, void *refCon );
```

DISCUSSION

Installs a window content paint proc. To remove a previously-installed paint proc (returning to the standard window manager erase-to-white content painting), pass `NULL` in the `paintProc` and `refCon` parameters.

```
typedef OptionBits WindowPaintProcOptions;
```

```
enum { kStandardPaintProcOptions = 0 // no paint proc options defined yet
};
```

VERSION NOTES

Available from CarbonLib 1.0 forward, running on Mac OS 8.7 and later.