Technical Note TN2058
The Font Panel for Carbon API

CONTENTS

This Technote describes the API used to display and interact with the Font Panel from a Carbon application on Mac OS X.

While most Carbon applications allow users to select fonts using a Fonts menu (preferably built and maintained using the Standard Font Menu API), users of Cocoa applications select fonts using the standard Font Panel floating dialog. This dialog is implemented by the NSFontPanel class using Objective C and Cocoa and is not directly accessible from Carbon applications. The Font Panel for Carbon API provides access to the Font Panel for Carbon applications on Mac OS X.

This Note is directed at application developers who want to allow their Mac OS X Carbon applications to use the Font Panel as a way for users to specify font family, typeface, and size settings for text.

[Sep 05 2002]

Using the Font Panel in Carbon

This section presents a brief overview of the use of the Font Panel from a Carbon application; detailed information about the API follows.

The user opens the Font Panel by selecting an appropriate menu item (such as `Font Panel&hellip;` from a `Fonts` menu), clicking a button, or via some other mechanism. Your application, in response, should generate a Carbon Event Manager command event with `kHICommandShowHideFontPanel`. The standard Carbon application event handler will detect the event and respond to it, calling `ShowHideFontPanel`.

> **Note:**
> If your application does not generate Carbon Event Manager command events, it can open the Font Panel by calling `ShowHideFontPanel` directly in response to the user's selection of a menu item or other HI element.

When a Carbon Event target (typically a control or window) gains the focus, your application calls `SetFontInfoForSelection`, providing the Font Panel with style run information for the currently selected text. `SetFontInfoForSelection` also lets your application specify the event target to which Font Panel-related Carbon events should be sent.

Whenever the selection changes in the focus, or if the style runs are changed programmatically, your application calls `SetFontInfoForSelection`. If the Font Panel is visible when this function is called, its contents are updated to reflect the style run information passed to the Font Panel; if the Font Panel is not visible, there is no user-visible effect, although the information supplied by `SetFontInfoForSelection` is saved so that, when the Font Panel becomes visible again, the correct settings will be displayed.

If your application has called SetFontInfoForSelection before the Font Panel is opened, the font selection information you supplied is provided to the Font Panel and is displayed when it opens; otherwise, the Font Panel will open with no selection.

As the user selects font settings from the Font Panel, your application receives font selection Carbon Events from the Font Panel. The settings selected by the user in the Font Panel are passed as event parameters in the kEventFontSelection event; your application simply extracts as many of the parameters as it can from the event and applies the font settings as it sees fit.

When the user closes the Font Panel, a Carbon event notifies your application that the window has closed.

> **Note:**
> Even if your Carbon application still uses the old WaitNextEvent-style event loop, it must install Carbon Event handlers to support the Font Panel. In particular, it has to handle Font Panel closed and selection events described in more detail below.

Back to top

The Font Panel API

SetFontInfoForSelection

Your application calls SetFontInfoForSelection to supply information to the Font Panel about the style runs in any currently selected text, so that the Font Panel can be updated to reflect this information. Multiple runs may be specified.

```
OSStatus SetFontInfoForSelection(
 OSType iStyleType,
 UInt32 iNumStyles,
 void* iStyles,
 HIObjectRef iFPEventTarget);
```

SetFontInfoForSelection accepts style run data in various formats; iStyleType specifies the format. Currently two values are supported:

- kFontSelectionATSUIType indicates that style run information will be specified using ATSUStyle collections.
- kFontSelectionQDType indicates that style run information will be specified using FontSelectionQDStyle records.

iNumStyles is the number of unique style runs being specified in the function call. If iNumStyles is 0, the Font Panel settings are cleared.

iStyles is a pointer to an array of style run information. If iStyleType is kFontSelectionATSUIType, iStyles points to an array of ATSUStyle collections. If iStyleType is kFontSelectionQDType, iStyles points to an array of FontSelectionQDStyle records.

iFPEventTarget is a reference to the Carbon Event Manager human interface object to which subsequent Font Panel events will be sent. This should be the window or control holding the current focus, or the application itself. The value can change from one call to another, as the user focus shifts. If this value is NULL, the Font Panel will send events to the application target as returned by GetApplicationEventTarget.

The following error conditions are returned:

- paramErr is returned if iNumStyles is nonzero but iStyles is NULL, or if it is detected that iNumStyles is not equal to the number of style runs specified.
- fontSelectionStyleErr is returned if any of the supplied style run information elements does not contain the minimum information required for Font Panel selection.

If SetFontInfoForSelection returns an error condition and the Font Panel is visible, the selection in the font panel will remain unchanged.

Your application is responsible for releasing any memory associated with the style run information pointed to by iStyles, following return from SetFontInfoForSelection.

When you call SetFontInfoForSelection, the Font Panel will scan the array of style run information and update the Font Panel as appropriate. If iNumStyles is 0, iStyles is assumed to be NULL and all Font Panel selections are

cleared.

> **Note:**
> Currently, the Font Panel selects no items at all if `iNumStyles` is greater than 1. However, your application should supply information for all style runs so that, if this behavior changes in the future, you will not have to modify your application.

You can call `SetFontInfoForSelection` even when the Font Panel is not open or visible. If you call it before the Font Panel is opened or while the Font Panel is hidden, when it later becomes visible the Font Panel will display the style information specified in the most recent call to `SetFontInfoForSelection`.

Back to top

ATSUStyle

When `ATSUStyle` objects are used to specify style run information, your application can call `ATSUSetAttribute` to set the font attributes in each `ATSUStyle`. At a minimum, your application must supply the following data in each `ATSUStyle`:

- `kATSUFontTag` (type `ATSUFontID`): The `ATSUFontID` of the selected text.
- `kATSUFontSize` (type `Fixed`): The size of the selected text, in points.

A pointer to an array of `ATSUStyle` objects is passed in the `iStyles` parameter in the call to `SetFontInfoForSelection`. Your application is responsible for calling `ATSUDisposeStyle` on each allocated `ATSUStyle` object after `SetFontInfoForSelection` returns.

Listing 1. Calling `SetFontInfoForSelection` using an `ATSUStyle`.

```
...
if (FPIsFontPanelVisible())
{
    OSStatus          status;
    ATSUStyle            aStyle;
    ATSUAttributeTag     atsuiTag[] = {kATSUFontTag, kATSUSizeTag};
    ByteCount            atsuiSize[] = {sizeof(ATSUFontID), sizeof(Fixed)};
    ATSUAttributeValuePtr   atsuiValue[2];

    status = ATSUCreateStyle(aStyle);

    /*
    Populate the attribute value array with pointers to the attribute values,
    in the correct order.
    */
    atsuiValue[0] = &atsuiFontID;
    atsuiValue[1] = &atsuiSize;

    /*
    Set the attributes in the ATSUStyle and send the info to the Font Panel.
    */
    status = ATSUSetAttributes(aStyle, 2, atsuiTag, atsuiSize, atsuiValue);
    status = SetFontInfoForSelection(kFontSelectionATSUIType, 1, &aStyle, NULL);

    /*
    Don't forget to release the ATSUStyle!
    */
    status = ATSUDisposeStyle(aStyle);
}
...
```

Back to top

FontSelectionQDStyle

If your application uses Quickdraw-style data types, you may want to specify style run information using `FontSelectionQDStyle` records:

```
typedef struct
{
    UInt32 version;
    FMFontFamilyInstance instance;
    FMFontSize size;
    Boolean hasColor;
    RGBColor color;
}
FontSelectionQDStyle;

const UInt32 kCurrentFontSelectionQDStyleZero = 0;
```

`version` is the current version number of the record, which should be set by your application to the appropriate constant defined in the header file, currently `kCurrentFontSelectionQDStyleZero`.

`instance` and `size` supply the Quickdraw font family reference, style specification, and typeface size.

If `hasColor` is true, `color` defines the color of the selected text. Otherwise, `color` is ignored.

Note:
Currently, color is not supported by the Font Panel for Carbon applications.

When `FontSelectionQDStyle` records are used to specify style information, the application must supply the family, style, and size of each run in the selection, as shown in Listing 2.

Note:
Currently, Quickdraw-specific styles such as outline and shadow are not supported by the Font Panel.

Listing 2. Calling `SetFontInfoForSelection` using a `FontSelectionQDStyle`.

```
/*
Notify the Font Panel that the font has changed.
*/
qdInfo.version = kFontSelectionQDStyleVersionZero;
qdInfo.instance = fontFamilyInstance;
qdInfo.size = fontSize;
GetFontInfoFromWindow(window, NULL, &(qdInfo.size));
qdInfo.hasColor = false;

status = SetFontInfoForSelection(kFontSelectionQDType,
        1, &qdInfo, NULL);
```

Back to top

Handling Change of Focus

When the user focus shifts, the component relinquishing focus should call `SetFontInfoForSelection`, specifying 0 for `iNumStyles` and NULL for `iEventTarget`; this tells the Font Panel that its settings are to be cleared. The component receiving the focus then calls `SetFontInfoForSelection` to register itself as the new event target (even if `iNumStyles` is still 0).

Remember that the user focus is the part of your application's HI toward which keyboard input is directed; it can be a window, a control, or any other HI element. In this case, only those HI elements to which you wish Font Panel events to be sent need to know when the user focus shifts. (See the documentation for the Carbon Event Manager for more information on Carbon Events and user focus.)

For example, if your application supports multiple windows, you can install a Carbon Event handler for each window that

will catch `kEventWindowFocusAcquired` and `kEventWindowFocusRelinquish` events (class `kEventClassWindow`) and call `SetFontInfoForSelection` appropriately. Listing 3 gives you an idea of how this code might look.

Calling `SetFontInfoForSelection` from a user focus event handler.

```
...

/*
Carbon Event Handler declarations.
*/
pascal OSStatus MyEventHandler  (EventHandlerCallRef    nextHandler,
     EventRef            event,
     void *           userData);

EventHandlerUPP     gMyEventHandlerUPP = NULL;

/*
This array specifies the events which MyEventHandler() can handle.
*/
static const EventTypeSpec sMyEventHandlerEvents[] =
     {
     { kEventClassWindow, kEventWindowFocusAcquired },
     { kEventClassWindow, kEventWindowFocusRelinquish },
     };
...

/*
Install Carbon Event Handler on each window to support the Font Panel.
*/
gMyEventHandlerUPP = NewEventHandlerUPP(MyEventHandler);

InstallWindowEventHandler(window1Ref, gMyEventHandlerUPP,
    GetEventTypeCount(sMyEventHandlerEvents),
    sMyEventHandlerEvents, 0, NULL);

InstallWindowEventHandler(window2Ref, gMyEventHandlerUPP,
    GetEventTypeCount(sMyEventHandlerEvents),
    sMyEventHandlerEvents, 0, NULL);

...

pascal OSStatus
MyEventHandler      (EventHandlerCallRef    nextHandler,
               EventRef            event,
               void *           userData)
{
    UInt32      eventClass = GetEventClass(event);
    HICommand   command;
    WindowRef   thisWindow = NULL;
    OSStatus    status = eventNotHandledErr;

    switch ( eventClass )
    {
     case kEventClassWindow:
     {
      switch (GetEventKind(event))
      {
       case kEventWindowFocusRelinquish:
       {
           /*
           Clear the Font Panel settings.
           */
           status = SetFontInfoForSelection(kFontSelectionATSUIType,
```

```
                    0, NULL, NULL);
      }
      break;

      case kEventWindowFocusAcquired:
      {
          /*
          Tell the Font Panel that the owner of this event handler
          is the new target.
          */
          status = GetEventParameter(event, kEventParamDirectObject,
      typeWindowRef, NULL,
      sizeof(WindowRef), NULL,
      &thisWindow);

             status = SetFontInfoForSelection(
      kFontSelectionATSUIType, 0, NULL,
      GetWindowEventTarget(thisWindow));
         }
         break;
       }
     }
     break;
    }

    return (status);

}   // MyEventHandler
```

Back to top

Showing and Hiding the Font Panel

It is your application’s responsibility to provide an interface by which the user can activate and deactivate the Font Panel. Typically this will be a `Show Font Panel` or `Show Fonts&hellip;` item in a `Format` or `Fonts` menu. The keyboard equivalent for this item should be command-T.

Your applications may have a button or other mechanism for activating the Font Panel; how you implement Font Panel activation will depend on the needs of your application.

Back to top

kHICommandShowHideFontPanel

The Carbon Event Manager HICommand `kHICommandShowHideFontPanel` toggles the state of the Font Panel, making it either visible or invisible. Your application should cause selection of the Font Panel activation/deactivation interface element to generate the Carbon Event Manager command event. This can be done, for example, by adding an entry to the `'xmnu'` resource corresponding to your Font menu.

Listing 4. Part of an `'xmnu'` resource implementing the `kHICommandShowHideFontPanel` `HICommand`.

```
resource 'xmnu' (131, "Font menu")
{
    versionZero
    {
     {
         /* array ItemExtensions: 5 elements */
         ...
         /* [3] */
         dataItem
         {
          kHICommandShowHideFontPanel,
          kMenuNoModifiers,
          currScript,
          0,
          0,
          noHierID,
          sysFont,
          naturalGlyph
         },
         ...
     }
    }
};
```

The standard Carbon application event handler will detect the event and respond to it, calling `ShowHideFontPanel`, toggling the state of the Font Panel. (See **ShowHideFontPanel**, below).

This command has no parameters.

Back to top

ShowHideFontPanel

Calling `ShowHideFontPanel` displays the Font Panel if it is not currently displayed, and hides it if it is currently displayed.

```
    OSStatus ShowHideFontPanel(void);
```

Your application can call this function directly, as shown in Listing 5, if it does not support the `kHICommandShowHideFontPanel` HICommand.

The following error conditions are returned:

`fontPanelShowErr` is returned if, for unknown reasons, the Font Panel cannot be made visible. Specific error conditions will be returned if the reason can be determined (e.g., `memFullErr`).

Listing 5. Calling `ShowHideFontPanel` directly.

```
switch ( eventClass )
{
    ...
    case kEventClassCommand:
    {
     /*
     Extract the HICommand from the event and handle it.
     */
     GetEventParameter(event, kEventParamDirectObject,
                typeHICommand, NULL, sizeof(HICommand),
                NULL, &command);

     switch ( command.commandID )
     {
         ...

         case kHICommandShowHideFontPanel:
         {
          /*
          Toggle the Font Panel state.
          */
          status = FPShowHideFontPanel();
         }
         break;

         ...
     }
    }
    ...
}
```

Back to top

Closing the Font Panel

After the user closes the Font Panel, either by clicking on its close button or using an application-supplied human interface element (such as a Hide Font Panel menu item), the Font Panel sends a Carbon event to the event target your application specified in its most recent call to `SetFontInfoForSelection`. This allows your application to update any menu items or other controls whose state may have to change because the Font Panel has closed. Your application must have installed a Carbon event handler to detect this event.

The event has no parameters.

The event is of class `kEventClassFont` and of type `kEventFontPanelClosed`.

> **Note:**
> Even if your application is `WaitNextEvent`-based, you need to install a Carbon event handler to detect when the Font Panel has closed.

Back to top

IsFontPanelVisible

```
    Boolean IsFontPanelVisible(void);
```

`IsFontPanelVisible` returns true if the Font Panel is currently visible, and false otherwise. Your application can call it, for example, to determine the proper state of the user interface element that controls the display of the Font Panel (e.g., whether to enable or disable the Font Panel menu item).

Back to top

Font Panel Selection Event

When the user selects an item in the Font Panel, the Font Panel sends a Carbon event to the event target your application specified in its most recent call to `SetFontInfoForSelection`. The Carbon event contains parameters that describe the font settings selected by the user; your application extracts these parameters from the Carbon event and applies them, for example, selected text. Your application must have installed a Carbon event handler to detect this event.

> **Note:**
> Even if your application is `WaitNextEvent`-based, you must install a Carbon event handler to get the settings from the Font Panel.

The event is of class `kEventClassFont` and of type `kEventFontSelection`.

The event contains parameters reflecting the current Font Panel settings in all supported formats. Your application can obtain the parameters using the `GetEventParameter` function:

- `kEventParamATSUFontID, typeATSUFontID`: If present, specifies the font ID of the selected font.
- `kEventParamATSUFontSize, typeATSUSize`: If present, specifies the size of the font as a Fixed value.
- `kEventParamFMFontFamily, typeFMFontFamily`: If present, specifies the font family reference of the font.
- `kEventParamFMFontStyle, typeFMFontStyle`: If present, specifies the Quickdraw style of the font.
- `kEventParamFMFontSize, typeFMFontSize`: If present, specifies the size of the font as an integer.

The Font Panel will send this Carbon event to your application every time the user selects an item in the Font Panel. Note that a selection may not define a complete style specification: the user may select a font family, which need not include a typeface; or he may select a size, which may not specify family or face. Therefore, not all the parameters listed above need be present in the Carbon event. Your application must check for all parameters which it recognizes and be able to apply partial font specifications to the currently selected text or to its current font settings as appropriate.

Back to top

Summary

Modifying your Carbon application to use the Font Panel API is not difficult, and enhances the Mac OS X experience by giving your users an interface for font selection that is consistent with Cocoa applications and is easier and often more convenient than a Fonts menu.

Back to top

References

Apple Computer, Inc. (2002) Apple Type Services for Unicode Imaging.

Apple Computer, Inc. (2002) Carbon Event Manager.

Apple Computer, Inc. (2002) Programming Topic: Font Panel.

Back to top

Downloadables

Acrobat version of this Note (45K) Download

Back to top
