Technical Note TN2026
Multilingual Text Engine Frequently Asked Questions

CONTENTS

This document contains a list of frequently asked questions for the Multilingual Text Engine (MLTE).

[Jul 11 2001]

Availability

Q: *Where are the MLTE APIs available? In addition, Apple mentioned an API for creating edit text controls that use MLTE instead of TextEdit. The API referenced was `CreateEditUnicodeTextControl`. Where is this API available?*

A: The MLTE APIs are available on Mac OS X 10.0 and later, on any version of CarbonLib that supports Mac OS 8.6 and later, and on Mac OS 9.0 and later.

The Unicode Edit Text Control is available on X 10.0 and later. It is not available on CarbonLib.

Back to top

Managing the View and Destination Rectangles

Q: *How do I set and change the view and destination rectangles in an MLTE object?*

A: The `TXNNewObject` function sets both the view and destination rectangles when an object is created. The values depend on the input parameters as follows:

a) Case when iFrame is NULL (window case)
When the `kTXNAlwaysWrapAtViewEdgeMask` frame option is not used in creating the object, the view rectangle is set to the window bounds, and the destination rectangle is set to a default value. For CarbonLib and Mac OS 9, the default value is the currently selected paper size (Page Setup), and for Mac OS X the default value is a constant (currently).

When the `kTXNAlwaysWrapAtViewEdgeMask` frame option is used, the view rectangle is set to the window bounds, and the destination rectangle is set to the window bounds minus the width of the scroll bars (if any were requested).

b) Case when `iFrame` is a pointer to a valid rectangle (pane case)
The initial behavior is the same whether the `kTXNAlwaysWrapAtViewEdgeMask` is turned on or off. The view rectangle (which includes the scroll bars if scroll bars are requested when the object is created) is set equal to the input rectangle, and the destination rectangle is set equal to the input rectangle minus the width of the scroll bars (if any were requested).

Now, let's discuss the APIs that allow MLTE clients to modify the view and destination rectangles.

The standard behavior for the function `TXNSetFrameBounds` is to set only the view rectangle, leaving the destination rectangle unchanged. However, if `kTXNAlwaysWrapAtViewEdgeMask` is used then the view and destination rectangles are kept in synch, so the size of both rectangles will be set.

The function `TXNResizeFrame` sets both the view and destination rectangles. The width and height of the view rectangle are set to the input width and height, respectively. The width of the destination is set to the input width minus the width of the scroll bar, if the vertical scroll bar is present. The height is not changed since the MLTE model assumes a destination rectangle with an infinite height.

The `TXNGETVIEWRECT` function retrieves the current view rectangle. Note that the rectangle includes the scroll bar areas, if any.

MLTE currently does not provide a function to set or retrieve the current destination rectangle. However, if you need to set the view and destination rectangles to different values you could call `TXNResizeFrame` and then `TXNSetFrameBounds`. The `TXNResizeFrame` function will set both the destination and the view rectangles, and `TXNSetFrameBounds` will set the bounds of the view rectangle.

Now, let's discuss in more detail the `kTXNAlwaysWrapAtViewEdgeMask` option. This option forces the view and the destination rectangles to be in synch (they are slightly different because of the scroll bars) so that, for instance, if a window is resized when this mask is used, the layout of the text will be recalculated. This option is useful, for instance, when an MLTE object spans a full window and you'd like the text to always wrap at the edge of the window.

Back to top

Using Quartz with MLTE

Q: *I'd like MLTE to render anti-aliased text on Mac OS X that looks like the rest of the system. Does MLTE support this feature?*

A: MLTE supports Quartz anti-aliased text on Mac OS X 10.0 and later. To do so, the client creates a CG context and passes it to MLTE using the `TXNSetTXNObjectControls` function. The following sample code turns on Quartz rendering:

```
TXNControlTag  controlTags[] = { kATSUCGContextTag };
 TXNControlData  controlData[1];
 CGContextRef  cgContext;

 if ( CreateCGContextForPort( aGrafPort, &cgContext ) == noErr )
 {

  controlData[0].uValue = (UInt32) cgContext;
  status = TXNSetTXNObjectControls(
    theMLTEObject,
    false,
    sizeof( controlTags ) / sizeof( TXNControlTag ),
    controlTags,
    controlData );
 }
```

Back to top

Cursor Visibility

Q: *When an MLTE object is not focused (`TXNFocus(txnObject, false)`), the cursor appears as a dotted line. Other text engines I've worked with hide the cursor when an object does not have focus. How do I do this with MLTE?*

A: On Mac OS X 10.0 and later, and on CarbonLib 1.2 and later, MLTE introduced several frame options to give MLTE clients more flexibility on the look of the cursor and text selection when an object does not have focus. The frame options are set when the object is created using TXNNewObject function.

The frame option
`kTXNDontDrawCaretWhenInactiveMask` indicates that the cursor shouldn't be drawn when the text object doesn't have focus, while the frame option `kTXNDontDrawSelectionWhenInactiveMask` indicates that the selection (if there is one) shouldn't be drawn when the text object doesn't have focus.

By setting these options when creating an MLTE object you will get the cursor to hide when the object does not have focus.

In addition, the control tags `kTXNDrawCaretWhenInactiveTag` and `kTXNDrawSelectionWhenInactiveTag`, and the corresponding values `kTXNDontDrawCaretWhenInactive`, `kTXNDrawCaretWhenInactive`, `kTXNDontDrawSelectionWhenInactive`, `kTXNDrawSelectionWhenInactive` were introduced to be used with the `TXNSetTXNObjectControls` function. The control tags and their corresponding values can be used to modify the look of an inactive caret and/or selection once an object has been created.

Back to top

Setting Text Style Attributes

Q: *How do I set the color, size and style of the current selection?*

A: With the `TXNSetTypeAttributes` function you can set one or more text attributes in a single call. The following code snippet sets the text in the current selection to be bold, red and of size 36.

```
// Create type attribute data structure
RGBColor  fontColor = { 0xFFFF, 0, 0 };
UInt32   fontSize = 36 << 16; // needs to be in Fixed format
TXNTypeAttributes attributes[] = {
        { kTXNQDFontStyleAttribute, kTXNQDFontStyleAttributeSize, bold },
        { kTXNQDFontColorAttribute, kTXNQDFontColorAttributeSize, &fontColor },
       { kTXNQDFontSizeAttribute, kTXNQDFontSizeAttributeSize, fontSize }
};

// Call TXNSetTypeAttributes
status = TXNSetTypeAttributes( myMLTEObject, 3, attributes,
     kTXNUseCurrentSelection,
     kTXNUseCurrentSelection );
```

Note that the `TXNSetTypeAttributes` function is called with the meta offset `kTXNUseCurrentSelection`; however, you can call this function for any range of valid offsets. In addition, note that by calling the function with only one attribute you could individually set the font style, the color and the font size.

Back to top

Setting the Font

Q: *How do I set the font of the current selection?*

A: First, you need to get an `ATSUFontID` for the font you are interested in using, then you pass it to MLTE using the `TXNSetTypeAttributes` function. For example, if you'd like to display the current selection in an MLTE object in "Apple Chancery", you can:

```
//Get the ATSUI font id
 Str255 fontName  ="\pApple Chancery";
 ATSUFontID   fontID;

 status = ATSUFindFontFromName(
   (Ptr) fontName+1,
   (long) fontName[0],
   kFontFullName,
   kFontNoPlatform,
   kFontNoScript,
   kFontNoLanguage,
   &fontID );

 //Pass the ATSUI font id to MLTE using the TXNSetTypeAttributes function
 TXNTypeAttributes     attributes[] =
      {kATSUFontTag, sizeof(ATSUFontID), {0}};

 attributes.data.dataValue = fontID;
 status = TXNSetTypeAttributes( myMLTEObject, 1, attributes,
    kTXNUseCurrentSelection,  kTXNUseCurrentSelection );
```

Additional documentation on the ATSUI APIs is available on the Apple Developer web site at <http://developer.apple.com/>.

Back to top

Getting the Font

Q: *How do I find out in MLTE what the font for a given style run is?*

A: You should use the `TXNGetContinuousTypeAttributes` function to find out the ATSUI font ID for a style run. Here is an example:

```
ATSUFontID   fontID;
 TXNContinuousFlags     continuousFlags;
 TXNTypeAttributes     attributes[] =
      {kATSUFontTag, sizeof(ATSUFontID), {0}};

 TXNGetContinuousTypeAttributes( myMLTEObject,
  &continuousFlags, 1,  attributes);

 fontID = attributes[0].data.dataValue;
```

Back to top

Scroll bars Rendering

Q: *I am creating an MLTE pane object (i.e., initial rectangle for the object is specified) and the scroll bars are not rendered properly when I call `TXNResizeFrame`. More precisely, the inside part of the scroll bar has a borderline but the outside does not. In addition, the top of the scroll bar leaves artifacts from the previous size. Do you have any suggestions?*

A: You should call the `InvalWindowRect` function to invalidate the old view rectangle used by the MLTE object and its scroll bars to avoid the problem you are mentioning. If you will use `TXNGETVIEWRECT` to get the view rectangle for the object remember that the rectangle this function returns does not include the outside border of either of the scroll bars. So, in this case, you first should grow it by one pixel all around by calling `InsetRect( &myMLTEObjectViewRect, -1, -1)`, and then call `InvalWindowRect` with the enlarged rectangle.

You should use the same approach if you are using TXNSetFrameBounds with a pane object.

Back to top

Carbon Applications

Q: I am Carbonizing an application that uses MLTE. Are there API call differences?

A: You cannot call `TXNTSMCheck` in a Carbon application. This is because TSM (Text Services Manager) is automatically initialized in Carbon. Also, there is no need for you to call `TXNIdle` or `TXNKeydown`. On Carbon, MLTE installs a timer to blink the cursor so your application does not need to call the `TXNIdle` function. In addition, TSM sends each keydown directly to the MLTE UnicodeForKeyEvent handler so your application will not receive them when a TXNObject has focus.

In Mac OS 10.0 and CarbonLib 1.3, the data structure `TXNCarbonEventInfo` was introduced to support MLTE Carbon event handlers. With the introduction of this functionality, MLTE Carbon applications can take full advantage of the Carbon event model for Text Input events.

Back to top

Carbon Events

Q: *How do I turn on support for Carbon events in MLTE?*

A: By default MLTE uses the Apple Event Manager for text input. However, with the recent addition of the `TXNCarbonEventInfo` data structure, MLTE clients can specify that MLTE use Carbon events to handle text input. To set up MLTE to support Carbon events for an MLTE object, you take the following steps:

a. Build a dictionary of event targets
b. Provide your own action key mapping callback function (optional)
c. Instantiate a `TXNCarbonEventInfo` structure with information about the specific events you want MLTE to handle
d. Call the function `TXNSetTXNObjectControls`

The following code turns on MLTE Carbon event handlers for Text Input events:

```
// Declare the variables
 TXNControlTag  controlTags[] = { kTXNUseCarbonEvents };
 TXNControlData  controlData[1];
 TXNCarbonEventInfo carbonEventInfo;
 CFStringRef   keys[] = { kTXNTextInputHandlerKey };
 EventTargetRef  values[] =
      { GetWindowEventTarget( myWindowRef ) };


 // Initialize the TXNCarbonEventInfo data structure for handling
 // Carbon Text Input events
 carbonEventInfo.useCarbonEvents = true;
 carbonEventInfo.filler = 0;
 carbonEventInfo.flags = kTXNNoAppleEventHandlersMask;
 carbonEventInfo.fDictionary =

 CFDictionaryCreate( kCFAllocatorDefault,
    (const void **) &keys,
    (const void **) &values,
    1,
   &kCFCopyStringDictionaryKeyCallBacks,
   NULL );

 controlData[0].uValue = (UInt32) &carbonEventInfo;

 // Tell MLTE to install its Carbon handlers
 status = TXNSetTXNObjectControls(
   theMLTEObject,
   false,
   sizeof( controlTags ) / sizeof( TXNControlTag ),
   controlTags,
   controlData );

 // Release the dictionary
 CFRelease( carbonEventInfo.fDictionary );
```

Performance Tips

Q: *What is the optimal way to clear the content of a `TXNObject`?*

A: You should call `TXNSetData` with a NULL pointer to clear the content of an MLTE object.

Q: *I have `kTXNAlwaysWrapAtViewEdgeBit` turned off. Should `TXNResizeFrame` be as fast as `TXNSetFrameBounds`?*

A: When `kTXNAlwaysWrapAtViewEdgeBit` is turned off, `TXNResizeFrame` is not going to be as fast as `TXNSetFrameBounds` because `TXNResizeFrame` causes re-layout of text while `TXNSetFrameBounds` does not, and hence the performance difference. `TXNResizeFrame` sets both the view and destination rectangles causing re-layout of text, while `TXNSetFrameBounds` sets only the view rectangle bounds so text re-layout is not necessary.

Note that `kTXNAlwaysWrapAtViewEdgeBit` doesn't do anything for `TXNResizeFrame`. When you call `TXNResizeFrame`, the view and destination rectangles are reset so that the text will wrap at the edge of the view rect (minus the vertical scroll bar, if one)

Back to top

Downloadables

Acrobat version of this Note (60K) Download

Back to top