Section 6 - Layout Settings

Purpose

In this section you will learn about some of the MLTE API that facilitate text formating.

Objective

Upon completion of this section you will be able to do the following:

- Use the MLTE object control API:
  TXNGetTXNObjectControls, TXNSetTXNObjectControls

Outline

The following topics will be covered in this section:

- word wrap
- line justification

Global Layout Settings

The sample application already provides a Format menu containing justification and word-wrap items. These kinds of settings are meant to apply to the entire TXNObject (i.e., the entire document).

Other items that you can set with MLTE on a global basis include line direction, tab values, read-only status, whether the caret is on or off, whether the bottom-line window is used, keyboard synchronization, and auto-indent. You can get and set these kinds of global characteristics with the **TXNGetTXNObjectControls** and **TXNSetTXNObjectControls** API.

```
EXTERN_API(OSStatus)TXNGetTXNObjectControls(TXNObject iTXNObject,
                                            ItemCount iControlCount,
                                            TXNControlTag iControlTags[],
                                            TXNControlData oControlData[]);


EXTERN_API(OSStatus)TXNSetTXNObjectControls( TXNObject iTXNObject,
                                            Boolean iClearAll,
                                            ItemCount iControlCount,
                                            TXNControlTag iControlTags[],
                                            TXNControlData iControlData[]);
```

The **iControlCount** parameter holds the number of TXNControlTag records in the array.

The **iControlTags** parameter is an array that holds the kind of formatting you wish to get or set. Constants for TXNControlTag are defined in MacTextEditor.h., such as kTXNTabSettingsTag and kTXNAutoIndentStateTag.

The **oControlData** parameter is an array of TXNControlData structures which are filled out with the information that was(get) or will be (set) requested via the iControlTags array. Constants for TXNControlData are defined in MacTextEditor.h.

In addition, the TXNSetTXNObjectControls function provides for an **iClearAll** parameter. This provides an easy way to reset all controls to the default, e.g., justification will be set to LMTESysJust, line direction will be set to GetSysDirection(), etc.

On systems which include ATSUI, all the ATSUI Line Control Attribute Tags can be passed to this function as a TXNControlTag. This is the case for all the ATSUI tags except kATSULineRotationTag. ATSUI Tags are applied to the entire TXNObject.

Our sample app has a DoWordWrap function. At Step 32 use the TXNGetTypeAttributes and TXNSetTypeAttributes functions to toggle the state of word wrapping in your document. Your code might look something like this:

```
controlTag[0] = kTXNWordWrapStateTag;
GetWindowProperty(window, 'GRIT', 'tObj', sizeof(TXNObject), NULL, &object);
status = TXNGetTXNObjectControls(object, 1, controlTag, controlData);

if (controlData[0].uValue == kTXNAutoWrap) // if we are autowrapped
{
   controlData[0].uValue = kTXNNoAutoWrap;      // toggle to not autowrapped
   CheckItem(layoutMenu, iAutoWrap, false);     // uncheck this menu item
}
else
{
   controlData[0].uValue = kTXNAutoWrap;
   CheckItem(layoutMenu, iAutoWrap, true);      // check this menu item
}
status = TXNSetTXNObjectControls(object, false, 1, controlTag, controlData);
```

Our sample app has a DoJustification function. At Step 33 use the TXNGetTypeAttributes and TXNSetTypeAttributes functions to change the state of justification in your document. Your code might look something like this:

```
controlTag[0] = kTXNJustificationTag;

GetWindowProperty(window, 'GRIT', 'tObj', sizeof(TXNObject), NULL, &object);
status = TXNGetTXNObjectControls(object, 1, controlTag, controlData);

if (controlData[0].sValue != justification) // if we have a new justification
{
   controlData[0].sValue = justification;
   status = TXNSetTXNObjectControls(object, false, 1, controlTag, controlData);
}
```

Notes

Bonus round

Our application has File menu items for PageSetup and Print. The relevant MLTE API are:

```
EXTERN_API(OSStatus)TXNPageSetup(TXNObject iTXNObject);

EXTERN_API(OSStatus)TXNPrint(TXNObject iTXNObject);
```

Use the TXNPageSetup and TXNPrint API at Step 34 and 35, respectively.