SGML Notes

This note contains various aspects of FDK programming with FrameMaker+SGML

Version 1.10

Copyright by Jesper Storm Bache, April 1998.
jesper@bache.com

Latest version always found at www.bache.com/sgml.html

See section for 1.2 info. about “price”

1.1 Introduction

This document focus on issues related to the development of *FDK* modules for FrameMaker+SGML.

Warning! THE USE OF THIS INFORMATION IS AT YOUR OWN RISK.

1.2 Price

The module is “Helpware”. If you find it useful please donate $5 - $10 to your favorite grassroot organization and send a mail to donations@bache.com. Subject: HelpWare. Contents: amount and organization.

1.3 The SGML hierarchy

FrameMaker+SGML represent the element hierarchy using a reduction algorithm. This algorithm merges markup and PCData nodes for container elements that only contains text.

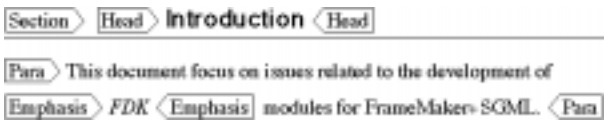
FrameMaker only generates separate PCData nodes if an element contains mixed contents (i.e. a mixture of elements and text).

This is illustrated in figure 1. Note that *Head* and *Emphasis* does not have a separate PCData node.

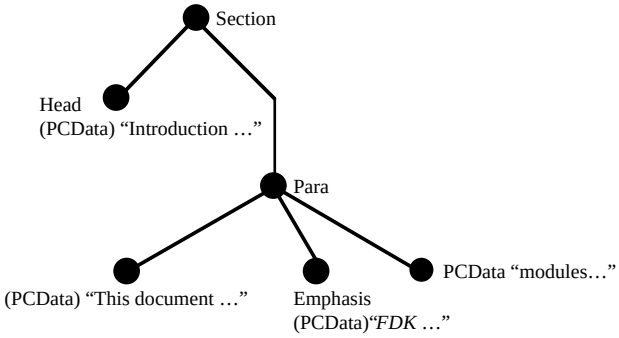
Because of this reduction algorithm you should traverse the document hierarchy slightly different than using a normal NLR algorithm. The figure shows a an example of a traversing algorithm. It makes use of the undocumented element type *FV_FO_UNSPECIFIED*. This element type is assigned to all PCData nodes that are used in mixed contents. The case you have to watch out for is the case of an element container with no children. This element is a *merged* element, and represent both a container element and a PCData node.

FIGURE 1 Reduction algorithm used to represent markup and PCData nodes

SGML formatted text in FrameMaker+SGML



Internal representation




```
Travers(inNode) {
        switch (type of inNode) {
                FV_FO_UNSPECIFIED:
                        this is a PCData node;
                FV_FO_CONTAINER:
                        if (num of children=0)
        /*NB*/                This is both a markup- and a
                                        PCData node
                        else
                                foreach child
                                        Travers(theChild);

                FV_FO_SYS_VAR
                FV_FO_XREF
                FV_FO_MARKER
                FV_FO_GRAPHIC
                FV_FO_EQN
                FV_FO_FOOTNOTE
                FV_FO_TBL
                FV_FO_TBL_TITLE
                FV_FO_TBL_HEADING
                FV_FO_TBL_BODY
                FV_FO_TBL_FOOTING
                FV_FO_TBL_ROW
                FV_FO_TBL_CELL
                FV_FO_RUBI_GROUP
                FV_FO_RUBI
                FV_FO_NUMTYPES
                        do something
        }
}
```

1.4 How to add EDD formatted text to a new element

The information in this section is valid for Frame version 5.1.x. I haven't tested if the described problem is still present in the FDK version 5.5.

When using F_ApiAddText() inside a newly created element the text that you insert is *not* formatted using the EDD. The text is inserted as unstyled text. In order to obtain the correct text formatting you have to "Import element definitions", where you import the definitions from the current document.

Importing element definitions reformats the whole document. This takes time and is not usable in an interactive mode.

This problem is considered a bug in the FDK, and it has been reported to Adobe.

The problem is caused by the fact that F_ApiAddText uses the textstyle to the immediate right of the insertion point. When you have created a new element there is no text inside the element, and hence no usable textstyle to the immediate right of the insertion point.

I have come up with the following work-around. It is based on the following scheme:

- Create a new element using the Api call
- We know that when the user enters text it is formatted correctly. The solution is therefore to simulate that the user enters a character. This is done by inserting a blank using *fcodes*.

- Now insert the text using F_ApiAddText to the immediate left of the previously added blank.

The code to do it looks something like:

FIGURE 2 How to insert EDD formatted text in a newly created element

```
// Find definition for the element
F_ObjHandleT theElementDef= F_ApiGetNamedObject(mDocId,
FO_ElementDef, StringT("Para"));

// Create the element
if (!theElementDef) {
        F_Printf(NULL, "The tag %s is not defined\n",theMarkup);
        F_ApiDeallocateString(&theMarkup);
        return;
}
F_ObjHandleT theNewObject = F_ApiNewElement(mDocId, theElementDef,
&tl);

// Append attributes if you want

// Insert a blank.
F_TextRangeT tr;
tr = F_ApiGetTextRange(mDocId, theNewObject, FP_TextRange);
tr.beg.offset++;
tr.end = tr.beg;
F_ApiSetTextRange(FV_SessionId, mDocId,FP_TextSelection, &tr);
static IntT fcodes[] = {' '};
F_ApiFcodes(sizeof(fcodes)/sizeof(IntT), fcodes);

// Add text (or insert children)
F_ApiAddText(...)

// remove the blank
tr = F_ApiGetTextRange(mDocId, theNewObject, FP_TextRange);
F_TextRangeT theBlank = tr;
theBlank.beg=theBlank.end;
theBlank.beg.offset--;
F_ApiDeleteText(mDocId, &theBlank);
```

1.5 Using Insert Element in hierarchy

Inserting elements with the function F_ApiNewElementInHierarchy() uses F_ElementLocT to identify the place where to insert the new element.

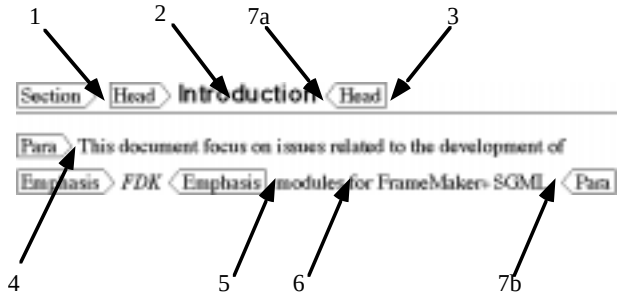
F_ElementLocT should be used in the following way:

- parentId is always the ID of the immediate parent container.
- childId depends on whether the element in inserted inside a PCData-node (with text on both sides) or next to an element start or element end.
  - If inserted in side text: childId is the id of the PCData node. This can be 0 if the markup node is reduces. (See section 1.3). Offset is number of characters to the left of the insertion point.
  - If not inserted inside text: childId is id of sibling to the right of the insertion point. Offset = 0. If the element is inserted at the end of an element then chil-did=0.

This algorithm basically says: ChildId is the id of the element to the right of the insertion point. Offset is the number of characters inside the child element. 0 is outside (to the left) of the element.

Examples of this is shown in figure 3.

FIGURE 3 Inserting a new element in hierarchy



NOTE: The cases 7a and 7b must be handled differently

1.5.1 Position 1

parentId = Id of Section

childId = Id of Head

offset = 0

The element is inserted next to an element start, so childId is Id of sibling immediately to the right and offset =0.

1.5.2 Position 2

parentId = Id of Head

childId = 0

offset = 4

The element is inside text. Offset = number of characters to the left of the insertion point. ChildId =0 because the element is reduces (has no PCData child node).

1.5.3 Position 3

parentId = Id of Section

childId = Id of Para

offset = 0

The element is inserted next of an element end or start, so childId is Id of sibling immediately to the right and offset =0.

1.5.4 Position 4

parentId = Id of Para

childId = Id of PCDatanode (This document…)

offset = 0

The element is inserted next of an element end or start, so childId is Id of sibling immediately to the right (this is a PCData node) and offset =0.

1.5.5 Position 5

parentId = Id of Para

childId = Id of PCDatanode (modules for…)

offset = 0

The element is inserted next of an element end or start, so childId is Id of sibling immediately to the right (PCData node) and offset =0.

1.5.6 Position 6

parentId = Id of Para

childId = Id of PCDatanode (modules for…)

offset = 7

The element is inserted inside text, so childId is Id of PCData node and offset = characters to the left of the insertion point.

1.5.7 Position 7a & 7b

Although 7a and 7b both are cases of “Insertion at the end of an element”, they must be handled diffenrently. The reason for this pitfall, is that FrameMaker+SGML use the “node reduction strategy” described in section 1.3. In 7a there is no PC-data node inside the “Head” markupnode.

This results in the following two *different* ways of insertion:

1.5.7.1 Position 7a

parentId = Id of Para

childId = 0

offset = 12

The element is inserted at the end of an element, so Right sibling = 0. NOTE THAT: Offset = 12, because “Head” only contains text, and because there are 12 characters in the element.

1.5.7.2 Position 7b

parentId = Id of Para

childId = 0

offset = 0

The element is inserted at the end of an element. Right sibling = 0 & offset = 0.

1.6 UID (persistent unique identifiers)

The FDK provides the function F_ApiGetUniqueObject(…) to get the persistent unique identifier (UID) for objects.

Just don’t use them. The UID’s of elements are *not* persistent and they do change. When the user chooses “undo” the UID of not affected elements can change!

Use attributes or the user-data field of the elements to keep track of elements.

1.7 Version history

1.7.1 Version 1.0

Initial draft