

1

What's New in FDK

This document lists the changes to the Frame[®] Developer's Kit (FDK) resulting from changes to the Adobe[®] FrameMaker[®] 9 product release.

What's New for Release 9

FrameMaker 9 has introduced the following changes that affect the FDK:

- Support for client's modeless dialogs in workspaces
- Support for client-owned cross-references
- Enhanced HTTP support
- New APIs
- New or changed FDK properties

Support for client's modeless dialogs in workspaces

With FrameMaker 9 providing support for workspaces, the client's modeless dialogs can become a part of a workspace. To make this work, the client has to handle the new notification *FA_Note_Dialog_Create*. This notification is sent to the client when the workspace has to launch a particular client's modeless dialog.

When the user closes a client's modeless dialog, the dialog event *FV_DlgClose* is issued. Now, with the support for workspaces, the dialog can also be closed due to workspace-related operations, such as switching workspaces. In such cases, instead of the dialog event *FV_DlgClose*, a new notification *FA_Note_QuitModelessDialog* is sent to the client. Therefore, the client must handle both of these events appropriately to achieve the desired result.

Also, within a workspace, if the dialog gets visible from a minimized/iconic state, then a dialog event *FV_DlgNeedsUpdate* is issued. This event indicates that the client's modeless dialog has become visible and should be updated, so that it does not display stale information.

Support for client-owned cross-references

With FrameMaker 9 providing support for client-owned cross-references, the client can use the new properties added to the *FO_XRef* object for identifying the cross-references it owns and handle them specifically, as required.

Also, a client can create its own dialog for cross-references. A new notification *FA_Note_DisplayClientXRefDialog* is sent to the client to display or update (in case it is already displayed) this dialog. If the client displays or updates its cross-reference dialog, then it sets the return value as *FR_DisplayedXRefDialog* using the API, *F_ApiReturnValue()* to indicate this to FrameMaker. If the return value is not set as explained, then FrameMaker assumes that the client did not display any dialog. Consequently, FrameMaker's standard cross-reference dialog is displayed.

Enhanced HTTP support

As a result of enhanced HTTP support in FrameMaker 9, behavior of some APIs and Properties have changed as described below.

F_ApiImport()

In FrameMaker 8, import-script-parameter *FS_UseHTTP* indicates that the graphic object's path is actually an HTTP URL. Hence, the object should be downloaded before importing it into the file.

FrameMaker 9 automatically determines whether the object-path is a local path or an HTTP path and acts accordingly. Although, a client can still set or reset this script parameter, the setting is ignored.

F_ChannelOpen()

This function works even if *FilePathT* supplied to *F_ChannelOpen()* API is prepared from an HTTP path. Because other channel-related functions operate on *ChannelT* that is outputted from *F_ChannelOpen()*, they are not impacted.

F_FilePathProperty()

This function works for HTTP paths as well. Property of the cached file is returned instead of the server file.

F_PathNameType()

This function returns *FDosPath* type for HTTP paths. For Windows, the API *F_PathNameType()* logically translates to the API *F_PathNameValid()* that returns *FDosPath*, if the path is valid.

F_FilePathToPathName() and F_PathNameToFilePath()

These functions also support HTTP paths for file path to path conversion and vice-versa.

FP_InsetURL and FP_InsetFile

For HTTP objects, the original file path of the object is its URL instead of the cached local path (as it used to be in FrameMaker 8).

In FrameMaker 8, using *F_ApiGetPropVal()* and *F_ApiGetSring()* for *FP_InsetFile* property for an object returns its local disk path for both non-HTTP objects and cached HTTP objects. And, using *F_ApiGetPropVal()* and *F_ApiGetSring()* for *FP_InsetURL* returns the URL string of the object if it has an HTTP path. For non-HTTP paths, *FP_InsetURL* would return NULL.

In FrameMaker 9, both these APIs yield the same result for the properties - *FP_InsetFile* and *FP_InsetURL*. The API functions return the disk path for non-HTTP objects and the URL string for HTTP objects. Similarly, using APIs *F_ApiSetPropVal()* and *F_ApiSetSring()* for both *FP_InsetFile* and *FP_InsetURL*, will set the original path of the object whether the string is the disk path or a URL.

Following are some more APIs that work with HTTP paths:

- *F_ChannelClose()*
- *F_ChannelEof()*
- *F_ChannelFlush()*
- *F_ChannelPeek()*
- *F_ChannelRead()*
- *F_ChannelSeek()*
- *F_ChannelSize()*
- *F_ChannelTell()*
- *F_ChannelWrite()*
- *F_FilePathBaseName()*
- *F_FilePathCopy()*
- *F_FilePathFree()*
- *F_FilePathParent()*
- *F_FilePathToPathName*
- *F_GetFilePath()*
- *F_PathNameTOFilePath*
- *F_Printf()*
- *F_ReadBytes()*
- *F_ReadLongs()*
- *F_ReadShorts()*
- *F_ResetByteOrder()*
- *F_Scanf()*
- *F_SetByteOrder()*

- F_WriteBytes()
- F_WriteLongs()
- F_WriteShorts()

Legacy clients that have not been recompiled with FDK 9

Most legacy clients that have been compiled with an earlier release of FDK will work with FrameMaker 9 without modification.

Recompiling old clients with FDK 9

For recompiling FDK 8 clients with FDK 9, the linking should be modified to include updated ICU, Xalan, and Xerces libraries.

Modified APIs

The behavior of following APIs has been modified and enhanced to provide better HTTP support in FrameMaker. See “Enhanced HTTP support” on page 2 for details.

- F_ApiImport
- F_ChannelClose()
- F_ChannelEof()
- F_ChannelFlush()
- F_ChannelOpen
- F_ChannelPeek()
- F_ChannelRead()
- F_ChannelSeek()
- F_ChannelSize()
- F_ChannelTell()
- F_ChannelWrite()
- F_FilePathBaseName()
- F_FilePathCopy()
- F_FilePathFree()
- F_FilePathParent()
- F_FilePathProperty

- `F_GetFilePath()`
- `F_PathNameType`
- `F_Printf()`
- `F_ReadBytes()`
- `F_ReadLongs()`
- `F_ReadShorts()`
- `F_ResetByteOrder()`
- `F_Scanf()`
- `F_SetByteOrder()`
- `F_WriteBytes()`
- `F_WriteLongs()`
- `F_WriteShorts()`

New APIs

F_ApiNewBookComponentOfTypeInHierarchy()

Inserts a book component of a specified type at a specified position in a structured FrameMaker book.

Synopsis

```
#include "fapi.h"
...
F_ObjHandleT F_ApiNewBookComponentOfTypeInHierarchy(F_ObjHandleT bookId,
    ConStringT compName,
    IntT compType,
    const F_ElementLocT *elemLocp);
```

Arguments

bookId	The ID of the book to add the component to.
compName	The name of the component.
compType	The type of book component to be created.
elemLocp	The position at which to add the new book component.

You can specify the following values for compType.

compType	Meaning
FV_BK_FOLDER	Folder type book component.
FV_BK_GROUP	Group type book component.

Returns

The ID of the new element that corresponds to the book component, or 0 if an error occurs.

If *F_ApiNewBookComponentOfTypeInHierarchy()* fails, the API assigns one of the following values to *FA_errno*.

FA_errno Value	Meaning
FE_BadBookId	Invalid book ID.
FE_BadParameter	Invalid compType.
FE_BadNew	The object can't be created.
FE_BookUnStructured	Specified book is unstructured.

Example

The following code adds a component of type folder to a book.

```
. . .
F_ObjHandleT bookId, elemId;
F_ElementLocT elemLoc;

/* Get ID of active book and its highest level element. */
bookId = F_ApiGetId(0, FV_SessionId, FP_ActiveBook);
elemLoc.childId = F_ApiGetId(FV_SessionId, bookId,
                             FP_HighestLevelElement);
elemLoc.parentId = 0;
elemLoc.offset = 0;

/* Insert the new element. */
elemId = F_ApiNewBookComponentOfTypeInHierarchy(bookId,
                                                "TechDoc Folder", FV_BK_FOLDER,
                                                &elemLoc);
. . .
```

F_ApiMoveComponent()

Moves a book component within the book.

Synopsis

```
#include "fapi.h"
. . .
VoidT F_ApiMoveComponent(F_ObjHandleT bookId,
                        F_ObjHandleT compId,
                        IntT moveAction);
```

Arguments

bookId	The ID of the book that contains the component.
compId	The ID of the book component that is to be moved.
moveAction	Specifies the action to move the component.

You can specify the following values for moveAction.

moveAction	Meaning
FA_COMPONENT_MOVEUP	Move the component up at the same hierarchical level.
FA_COMPONENT_MOVEDOWN	Move the component down at the same hierarchical level.
FA_COMPONENT_PROMOTE	Move the component to a higher/outer level in hierarchy.

moveAction	Meaning
FA_COMPONENT_DEMOTE	Move the component to a lower/inner level in hierarchy.

Returns

VoidT

Example

The following code moves up a book component.

```

. . .
F_ObjHandleT bookId;
F_ObjHandleT comp1Id, comp2Id;

/* Get ID of active book and its second component. */
bookId = F_ApiGetId(0, FV_SessionId, FP_ActiveBook);
comp1Id = F_ApiGetId(FV_SessionId, bookId,
                    FP_FirstComponentInBook);
comp2Id = F_ApiGetId(bookId, comp1Id,
                    FP_NextComponentInBook);

/* Move the component. */
F_ApiMoveComponent(bookId, comp2Id,
                  FA_COMPONENT_MOVEUP);
. . .

```

F_ApiUpdateXRef()

Updates a specified cross-reference in the document.

Synopsis

```

#include "fapi.h"
...
IntT F_ApiUpdateXRef(F_ObjHandleT docId,
                   F_ObjHandleT srcDocId,
                   F_ObjHandleT xrefId);

```

Arguments

docId	The ID of the document that contains the cross-reference.
srcDocId	The ID of the source document that the cross-reference references.
xrefId	The ID of the cross-reference to be updated.

Returns

FE_Success if it succeeds, or an error code if an error occurs.

If `F_ApiUpdateXRef()` fails, the API assigns one of the following values to `FA_errno`.

FA_errno Value	Meaning
<code>FE_BadDocId</code>	Invalid document ID.
<code>FE_BadXRefSrcDoc</code>	Invalid source document ID.
<code>FE_BadObjId</code>	Invalid cross-reference ID.
<code>FE_XRefUnresolved</code>	FrameMaker cannot update the cross-reference, as it cannot find the source.

Example

The following code updates the cross-reference to the same or another document.

```
. . .
F_ObjHandleT xrefId = F_ApiGetId(FV_SessionId, docId,
                                FP_FirstXRefInDoc);

F_ObjHandleT srcDocId;
StringT srcFile;

/* Open the source document */
srcFile = F_ApiGetString(docId, xrefId, FP_XRefFile);

if(F_StrIsEmpty(srcFile))
    srcDocId = docId;
else
{
    F_PropValsT params, *returnp = NULL;

    /* Set the open & return parameters */
    . . .

    srcDocId = F_ApiOpen(srcFile, &params, &returnp);
}

F_ApiUpdateXRef(docId, srcDocId, xrefId);
. . .
```

F_ApiApplyAttributeExpression()

Applies an attribute expression to a document to perform attribute-based-filtering.

Synopsis

```
#include "fapi.h"
...
IntT F_ApiApplyAttributeExpression(F_ObjHandleT docId,
    F_ObjHandleT attrExprId);
```

Arguments

docId	The ID of the document to which the attribute expression is to be applied.
attrExprId	The ID of the attribute expression to be applied to the document.

Returns

FE_Success if it succeeds, or an error code if an error occurs.

If *F_ApiApplyAttributeExpression()* fails, the API assigns one of the following values to FA_errno.

FA_errno Value	Meaning
FE_BadDocId	Invalid document ID.
FE_BadObjId	Invalid expression ID.
FE_InvalidAttrExpr	The attribute expression being invalid cannot be applied to the document.
FE_WrongProduct	The current FrameMaker product doesn't support the operation.
FE_SystemError	Couldn't allocate memory.

Example

The following code applies the expression named 'WebOutput' to the document.

```

. . .
F_ObjHandleT attrExprId = F_ApiGetNamedObject(docId, FO_AttrCondExpr,
                                             "WebOutput");
F_ApiApplyAttributeExpression(docId, attrExprId);
. . .

```

F_ApiNetLibSetAuthFunction()

Sets a callback function that is called for setting the username/password information before performing the NetLib related authentication.

Synopsis

```

#include "fapi.h"
...
VoidT F_ApiNetLibSetAuthFunction(VoidT (*p_auth_func) (ConStringT url,
StringT username, StringT password,
IntT *cancelp) );

```

Arguments

<code>p_auth_func</code>	Callback function that sets the username and password information for the NetLib related authentication.
--------------------------	--

NOTE:

After registering the callback function, whenever NetLib needs authentication, it calls this function with the server's URL as *'url'* parameter. This function checks the URL string and accordingly fills-in the username and password fields and sets **cancelp* to 0. If you want to cancel the authentication request from NetLib, set **cancelp* to 1. In this case, set the username and password to empty strings.

To unregister the callback function, call *F_ApiNetLibSetAuthFunction()* with NULL as the parameter. Then, FrameMaker's standard NetLib authentication dialog is displayed for the purpose.

The server authentication fails if the strings are set with an invalid username or password through this callback function. NetLib calls the function again and this time too, the function sets the wrong authentication information. This process will go on and FrameMaker will hang. Therefore, use some mechanism (like counters) in the callback function to avoid such a condition. For example, if the authentication requests for a particular server reaches a certain number of times (say 3), cancel the later requests for that server.

Returns

VoidT

Example

The following code sets the authentication callback function as *'SplAuthFunction()'*. The callback function matches the URL provided and fills in the username & password accordingly.

```

. . .
    F_ApiNetLibSetAuthFunction(SplAuthFunction);
. . .

. . .
VoidT SplAuthFunction(ConStringT url, StringT username,
                    StringT password, IntT *cancelp)
{
    static int attempts_srv1 = 0;
    static int attempts_srv2 = 0;

    // send authentication information for first three attempts only
    if (attempts_srv1 < 3 &&
        F_StrCmp(url, (ConStringT)"hostname:8080") == 0)
    {

        F_StrCpy(username, (ConStringT)"user123");
        F_StrCpy(password, (ConStringT)"pass234");
        *cancelp = 0;
    }
}

```

```
        attempts_srv1++;
    }
    else if (attempts_srv2 < 3 &&
        F_StrCmp(url, (ConStringT)"cmsserver") == 0)
    {
        F_StrCpy(username, (ConStringT)"Admin");
        F_StrCpy(password, (ConStringT)"Password");
        *cancelp = 0;
        attempts_srv2++;
    }
    else
    {
        F_StrCpy(username, (ConStringT) "");
        F_StrCpy(password, (ConStringT) "");
        *cancelp = 1;
        attempts_srv1 = 0;
        attempts_srv2 = 0;
    }
}
. . .
```

New and changed FDK components

The following sections describe the new or modified properties and property values for existing objects. Properties and values that have not changed from earlier releases are not listed here.

New Objects

None.

New Properties

FO_BookComponent Properties

FP_FirstComponentInBookComponent
FP_BookComponentParent
FP_ExcludeBookComponent
FP_BookComponentTemplatePath
FP_XmlApplicationForBookComponent
FP_BookComponentTitle
FP_ComponentType

FO_Doc Properties

FP_PDFGenerateForReview
FP_PrintDitavalFileName

FO_Fn Properties

FP_FnAnchorString

FO_XRef Properties

FP_XRefClientName
FP_XRefClientType
FP_XRefSrcElemNonUniqueId
FP_XRefAltText

New Property Value Constants

FP_ComponentType Values

FV_BK_FILE

FV_BK_FOLDER

FV_BK_GROUP

FS_FileType Values

FV_SaveFmtCompositeDoc

FV_SaveFmtBookWithXml

FV_SaveFmtBookWithFm

New Script Parameters

FS_DitavalFile

Deprecated Script Parameters

FS_UseHTTP

Modified Properties

FP_PDFConvertCMYKtoRGB (now available for windows)

FP_InsetURL

FP_InsetFile

New Notifications

FA_Note_Dialog_Create

FA_Note_DisplayClientXRefDialog

FA_Note_PreUpdateXRefs

FA_Note_PostUpdateXRefs

FA_Note_QuitModelessDialog

Modified Hint Strings

The hint string used for certain filters have been modified. The following table lists the new hint strings for FrameMaker 9. The hint strings that have not changed for FrameMaker 9 are not listed here.

Filter type	New Hint String
GIF	"0001AIDEGIF WIN3 "
JPEG	"0001AIDEJPEGWIN3 "
PNG	"0001AIDEPNG WIN3 "
PSD	"0001APSLPSD WIN3 "