1 What's New in FDK 10

This document lists the changes to the Frame® Developer's Kit (FDK) resulting from changes to the Adobe® FrameMaker® 10 product release.

What's new for release 10

FrameMaker® 10 has the following changes that affect the FrameMaker® Developer's Kit (FDK):

- Modeless dialog control support
- New or changed structures
- New or changed APIs
- New FDK properties
- New notifications

Modeless dialog control support

Client can now capture behavior of modeless dialog and control its position. For modeless dialogs, client can handle the dialog's show-hide behavior using the events FV_DlgHide and FV_DlgShow. It can also handle its close behavior by capturing event FV_DlgClose and control its hide behavior on close. Position of the modeless dialog can be controlled by DLG(.dlg file) and its docking can be controlled too.

The following table details the FDK support for modeless dialog:

Functionality	FDK support
Open a modeless dialog as docked or undocked	F_ApiSetInt(session_id,dialog_id, FP_DockDialog, Dock_value); Dock_value can be one of the following: FV_DIALOG_DOCK_NONE FV_DIALOG_DOCK_BOTTOM FV_DIALOG_DOCK_RIGHT FV_DIALOG_DOCK_LEFT
Determine if a dialog is docked or undocked	F_ApiGetInt(session_id, dialog_id,FP_IsDialogDocked);

Functionality	FDK support
Determine if a dialog is displayed	F_ApiGetInt(session_id, dialog_id, FP_IsDialogVisible);
Control dialog on hide/close behavior	Client can capture the FV_DlgClose event. To hide a dialog on close, use the following return value: F_ApiReturnValue(FR_HideDialogOnClose);
Control the position of the dialog	The position of the dialog can be controlled using .dlg file.
Capture the show and hide event	Events FV_DlgHide & FV_DlgShow

Legacy clients that have not been recompiled with FDK 10

Most legacy clients compiled with an earlier release of FDK work with FrameMaker 10 without modification.

Recompiling old clients with FDK 10

For recompiling FDK 9 clients with FDK10, change the Runtime Library flag (C/C++ -> Code Generation) to Multi-threaded DLL (/MD).

Recompiling old structured clients with FDK 10

For structured clients, you can get a compiler error, as we have changed the structure and some APIs.

Data Structures

The following structure is changed in fmstruct.lib:

- SwEventT

SwEventT

Describes a FrameMaker structure for an export event. SwEventT is modified to use the enhanced version of attributes structure F_AttributesExT.

```
typedef struct {
     SwEventTypeT evtype;
```

```
    F_TextLocT txtloc;
    F_ObjHandleT fm_elemid;
    F_AttributesExT fm_attrs; /* Changed the type of fm_attrs from
                               * F_AttributesT to F_AttributesExT*/
    F_ObjHandleT fm_objid;
    StringT text;
    UIntT charcode;
    StringT chartag;
} SwEventT;
```

APIs

The following APIs are changed:

- Sr_GetAttrVals()
- Sr_SetAttrVals()
- Sr_GetAttrVal()
- Sr_SetAttrVal()

Sr_GetAttrVals()

Retrieves the list of attribute values for the specified conversion object. The return type of attributes list is changed from `F_AttributesT` to `F_AttributesExT` as follows:

```
F_AttributesExT Sr_GetAttrVals(SrConvObjT srObj);
```

Sr_SetAttrVals()

Specifies the list of attribute values to use with the current conversion object. The new prototype is as follows:

```
SrwErrorT Sr_SetAttrVals(SrConvObjT srObj,
                         F_AttributesExT *attValList);
```

Sr_GetAttrVal()

Retrieves the attribute-value data structure for a single, specified attribute that is associated with the specified conversion object. The return type of attribute is changed from `F_AttributeT` to `F_AttributeExT` as follows:

```
F_AttributeExT Sr_GetAttrVal(SrConvObjT srObj,
                         StringT fmAttrName);
```

Sr_SetAttrVal()

Sets the attribute value for a named object associated with the current import conversion object. The new prototype is as follows:

```
SrwErrorT Sr_SetAttrVal(SrConvObjT srObj,
F_AttributeExT *attVal);
```

Data structures

The following structures are changed or added in FDK 10:

- F_TypedValsT
- F_TypedValT
- F_AttributesExT
- F_AttributeExT

F_TypedValsT

The F_TypedValsT structure describes a list of F_TypedValT.

```
typedef struct {
     IntT len;
     F_TypedValT *val;
} F_TypedValsT;
```

F_TypedValT

F_TypedValT specifies an individual property value. The structure is enhanced by adding F_TypedValsT, which now enables us to have a property with value as set of F_TypedValT.

```
typedef struct {
IntT valType; /* The type of value. See following table. */
union {
     StringT sval;              /* String value */
     F_StringsT ssval;          /* Set of strings */
     F_MetricsT msval;          /* Set of metrics */
     F_PointsT psval;           /* Set of points */
     F_TabsT tsval;             /* Set of tabs */
     F_TextLocT tlval;          /* Text location */
     F_TextRangeT trval;        /* Text range */
     F_AttributeDefsT adsval;
     F_AttributesT asval;
     F_AttributesExT asvalEx;         /* Only for Internal Use. */
     F_ElementCatalogEntriesT csval; /* Element Catalog */
     F_IntsT isval;                   /* Set of integers */
     F_UIntsT uisval;                 /* Set of unsigned integers */
```

```
    F_TypedValsT *valsval;          /* Set of F_TypedValT */
    IntT ival;                      /* integer */
    } u;
} F_TypedValT;
```

F_AttributesExT

The F_AttributesExT structure describes a list of F_AttributeExT.

```
typedef struct {
    IntT len;
    F_AttributeExT *val;
} F_AttributesExT;
```

F_AttributeExT

Describes an individual attribute. The new members added to the existing structure F_AttributeT are for internal purpose. Do not use the new members.

```
typedef struct F_AttributeExT
{
    StringT name;
    F_StringsT values;
    F_StringsT originalValues;      /* Internal purpose. */
    UByteT valflags;                /* validation error flags - R/O */
    UByteT allow;                   /* allow validation error as
                                    special case*/
    UByteT overriddenFlags;         /* Internal purpose. */
} F_AttributeExT;
```

New APIs

F_ApiCopyStructureTypes

The following function copies newly added structures. The function performs a deep copy and copies any arrays or strings referenced by the structure.

Synopsis

```
#include "fapi.h"
...
F_AttributesExT* F_ApiCopyAttributesEx(
                  F_AttributesExT *fromattributes);
F_AttributeExT* F_ApiCopyAttributeEx(
                  F_AttributeExT *fromattribute);
VoidT F_ApiDeallocateTypedVals(
                  F_TypedValsT *valsvalp);
```

Arguments

`fromStructureType`	The structure to copy.

Returns

A copy of the specified structure

F_ApiDeallocate*StructureType*()

The following function deallocates memory referenced by the newly added structures. The function performs a deep deallocation and deallocates any arrays or strings referenced by the structure.

Synopsis

```
#include "fapi.h"
...
VoidT F_ApiDeallocateAttributesEx(
                  F_AttributesExT *attributes);
VoidT F_ApiDeallocateAttributeEx(
                  F_AttributeExT *attribute);
VoidT F_ApiDeallocateTypedVals(
     F_TypedValsT *valsvalp);
```

Arguments

`StructureType`	The structure referencing memory that needs to be deallocated.

Returns

VoidT

F_ApiApplyAttributeExpressionAsCondition()

Applies an attribute-based expression to the document where the filtered text is converted to conditional text.

Synopsis

```
#include "fapi.h"
...
StatusT F_ApiApplyAttributeExpressionAsCondition(
     F_ObjHandleT docId,
     F_ObjHandleT attrExpId,
     F_ObjHandleT condId,
     BoolT removePreviouslyApplied);
```

Arguments

docId	The ID of the document on which attrExpId is to be applied
`attrExpId`	The ID of the attribute expression to be applied on the document
`condId`	The ID of the conditional text format that will be applied on the filtered text
`removePreviouslyApplied`	If true, then remove the conditional text settings at all locations in the document, whereever condId is applied.

Returns

If succeeds: `FE_Success`

If an error occurs: An error code

If API fails, the API assigns one of the following values to FA_errno:

FA_errno value	Meaning
FE_Success	Operation is successful
FE_BadDocId	Invalid document ID
FE_InvalidAttrExpr	The expression string set in the attrExpId's object is invalid
FE_WrongProduct	Current product interface isn't Structured FrameMaker
FE_BadObjId	Invalid object ID

F_ApiPreviewAttributeExpression()

Applies an attribute-based expression to the document and the filtered text is displayed in the specified color.

Synopsis

```
#include "fapi.h"
...
StatusT F_ApiPreviewAttributeExpression(
     F_ObjHandleT docId,
     F_ObjHandleT attrExpId,
     F_ObjHandleT colorId);
```

Arguments

docId	The ID of the document on which attrExpId is to be applied
`attrExpId`	The ID of the attribute expression to be applied on the document
`colorId`	The ID of the color that will be applied on the filtered text

Returns

If succeeds: `FE_Success`

If an error occurs: An error code

If API fails, the API also assigns one of the following values to FA_errno:

FA_errno value	Meaning
FE_Success	Operation is successful
FE_BadDocId	Invalid document ID
FE_InvalidAttrExpr	The expression string set in the attrExpId object is invalid
FE_WrongProduct	Current product interface isn't Structured FrameMaker
FE_BadObjId	Invalid object ID

F_ApiAddNewBuildExpr()

Adds a Boolean conditional expression to the document

Synopsis

```
#include "fapi.h"
...
IntT F_ApiAddNewBuildExpr(
     F_ObjHandleT docId,
     ConStringT exprName,
     ConStringT exprCondition);
```

Arguments

docId	The ID of the document in which the Boolean conditional expression will be added
`exprName`	The name of the Boolean conditional expression
`exprCondition`	The Boolean conditional expression string

Returns

If succeeds: `FE_Success`

If an error occurs: An error code

If API fails, the API also assigns one of the following values to FA_errno:

FA_errno value	Meaning
FE_Success	Operation is successful
FE_BadDocId	Invalid document ID
FE_ReadOnly	Document is read-only
FE_BadName	Boolean conditional expression string is invalid

F_ApiDeleteBuildExpr()

Adds a Boolean conditional expression to the document

Synopsis

```
#include "fapi.h"
...
IntT F_ApiDeleteBuildExpr(
     F_ObjHandleT docId,
     ConStringT exprName,
     ConStringT exprCondition);
```

Arguments

docId	The ID of the document in which the build expression will be added
`exprName`	The name of the build expression
`exprCondition`	The Boolean condition expression string

Returns

If succeeds: `FE_Success`

If an error occurs: An error code

If API fails, the API assigns one of the following values to FA_errno

FA_errno value	Meaning
FE_Success	Operation is successful
FE_BadDocId	Invalid document ID
FE_ReadOnly	Document is read-only
FE_BadName	Boolean conditional expression string is invalid

F_ApiSetActiveBuildExpr()

Applies the Boolean conditional expression to the document.

Synopsis

```
#include "fapi.h"
...
IntT F_ApiSetActiveBuildExpr(
     F_ObjHandleT docId,
     ConStringT exprName);
```

Arguments

docId	The ID of the document on which the Boolean conditional expression will be applied
`exprName`	The name of the Boolean conditional expression to be applied.

Returns

If succeeds: `FE_Success`

If an error occurs: An error code

If API fails, the API also assigns one of the following values to FA_errno

FA_errno value	Meaning
FE_Success	Operation is successful
FE_BadDocId	Invalid document ID
FE_ReadOnly	Document is read-only
FE_BadName	Boolean conditional expression string is invalid

F_ApiGetActiveBuildExpr()

The function returns the name of the active expression in the document. It returns null if no expression is active.

Synopsis

```
#include "fapi.h"
...
StringT F_ApiGetActiveBuildExpr(
     F_ObjHandleT docId);
```

Arguments

docId	The ID of the document

Returns

The name of the active expression in the document and null if none is active

If API fails, the API assigns one of the following values to FA_errno:

FA_errno value	Meaning
FE_Success	Operation is successful
FE_BadDocId	Invalid document ID

F_ApiGetBuildExpr()

The function returns the Boolean conditional expression in the document with the given name.

Synopsis

```
#include "fapi.h"
...
StringT F_ApiGetActiveBuildExpr(
     F_ObjHandleT docId,
     StringT exprName);
```

Arguments

docId	The ID of the document
exprName	The name of the Boolean conditional expression

Returns

The conditional expression with the given name and null if none is found

If API fails, the API assigns one of the following values to FA_errno:

FA_errno value	Meaning
FE_Success	Operation is successful
FE_BadDocId	Invalid document ID
FE_BadName	Invalid exprName

F_ApiGetBuildExprCatalog()

The function returns an array of all Boolean conditional expression names in the document.

Synopsis

```
#include "fapi.h"
...
F_StringsT F_ApiGetBuildExprCatalog(
     F_ObjHandleT docId);
```

Arguments

docId	The ID of the document

Returns

An array of all Boolean conditional expression names in the document

If API fails, the API assigns one of the following values to FA_errno:

FA_errno value	Meaning
FE_Success	Operation is successful
FE_BadDocId	Invalid document ID

F_ApiNetLibAuthenticateServer()

The function pre-registers a given server with the provided username and password, so that instead of prompting the user for authentication, the server uses the login information when required. This function is valid for the current session of FrameMaker.

Synopsis

```
#include "fapi.h"
...
StatusT F_ApiNetLibAuthenticateServer(
            ConStringT url,
            ConStringT username,
            ConStringT password);
```

Arguments

`url`	Fully qualified server URL
`username`	Username to authenticate
`password`	Password to authenticate

Returns

If succeeds: `FE_Success`

If fails: `FE_Transport`

F_ApiNetLibUploadFolder()

Uploads a folder and subfolders to the server.

Synopsis

```
#include "fapi.h"
...
StatusT F_ApiNetLibUploadFolder(
            ConStringT url,
            ConStringT localFolder);
```

Arguments

`url`	Fully qualified server URL
`localFolder`	Full path of the folder (on local disk) to be uploaded

Returns

If succeeds: `FE_Success`

If fails: `FE_Transport`

F_ApiConvertToText()

The function converts a locked text inset range to text.

Synopsis

```
#include "fapi.h"

...
StatusT F_ApiConvertToText(
            F_ObjHandleT docId,
            F_ObjHandleT textInsetId);
```

Arguments

`docId`	The ID of the document
`textInsetId`	The ID of the text inset to be converted

Returns

If succeeds: `FE_Success`

If fails: A non-zero integer

F_ApiTrackChangesAcceptAll ()

Accepts all the tracked changes in the specified document.

Synopsis

```
#include "fapi.h"

...
IntT F_ApiTrackChangesAcceptAll(
            F_ObjHandleT docId);
```

Arguments

`docId`	The ID of the document on which the Accept all command needs to be run

Returns

If succeeds: `FE_Success`

If fails: A non-zero integer

If the documents is not in Preview Off state: FE_AcceptRejectCalledOnWrongPreviewState (The Accept/Reject All operation does not take place)

F_ApiTrackChangesRejectAll ()

Rejects all the tracked changes in the specified document

Synopsis

```
#include "fapi.h"

...
IntT F_ApiTrackChangesRejecAll(
            F_ObjHandleT docId);
```

Arguments

`docId`	The ID of the document on which the Accept all command needs to be run

Returns

If succeeds: `FE_Success`

If fails: A non-zero integer

If the documents is not in Preview Off state: FE_AcceptRejectCalledOnWrongPreviewState (The Accept/Reject All operation does not take place)

F_ApiChooseFileExEx ()

Displays dialog boxes similar to FrameMaker's Open and Save dialog boxes. It displays directories and files in a list and allows the user to choose a file or directory. The new API provides enhancements to the existing F_ApiChooseFileEx function. The API allows the user to select multiple files and option is selected from the drop-down list of format string.

Synopsis

#include "fapi.h"

...

```
IntT F_ApiChooseFileExEx(F_StringsT *fileList,
                      IntT *formatIndp
                      StringT title,
                      StringT directory,
                      StringT stuffVal,
                      IntT mode,
                      StringT helpLink,
                      String formatStr);
```

Arguments

`fileList`	Reference to the empty F_StringsT. The function returns the list of selected files when you click OK.
`formatIndp`	Index of the selected item from the list of format strings.
`title`	The message that appears in the dialog box.
`directory`	The default directory when the dialog box is first displayed. If you specify an empty string, the last directory used by an FDK client is used. If no FDK client has used a directory, the directory specified by the session property, `FP_OpenDir`, is used.
`stuffVal`	The default value that appears in the input field when the dialog box first appears. If the dialog box type specified by `mode` doesn't have an input field, this string is ignored.
`mode`	A constant specifying the type of dialog box. See the table below for possible values.
`helpLink`	Obsolete for versions 6.0 and later; pass an empty string. The name of a document containing Help information for the dialog box or an empty string (`""`) if there is no Help document.
`formatStr`	Format string to be shown in the list.

You can also specify the following values for `mode`:

mode constant	Dialog box type
`FV_ChooseSelect`	Dialog box that allows the user to choose a file by clicking Select
`FV_ChooseOpen`	Dialog box that allows the user to choose a file by clicking Open
`FV_ChooseSave`	Dialog box that allows the user to select a file. It provides Save and Cancel buttons and an input field
`FV_ChooseOpenDir`	Dialog box that allows the user to choose a directory
`FV_ChooseMultiSelect`	Dialog box that allows the user to choose multiple files
FV_ChooseMultiOpen	Dialog box that allows the user to open multiple files

Returns

If the user clicked Open, Select, Use, or Save: 0

If the user clicked Cancel or an error occurred: A non-zero value

If `F_ApiChooseFile()` fails, the API assigns the following value to `FA_errno`.

FA_errno value	Meaning
`FE_Transport`	A transport error occurred

F_ApiDeleteUnusedFmts ()

Deletes unused formats (character, paragraph, or table) from the document.

Synopsis

```
#include "fapi.h"

...
IntT F_ApiDeleteUnusedFmts(
            F_ObjHandleT docId
            IntT objType);
```

Arguments

`docId`	The ID of the document for which unused formats are deleted
`objType`	The type of the object. If unused paragraph formats needs to be deleted, pass FO_PgfFmt. Similarly, pass FO_CharFmt or FO_TblFmt for character formats and table formats, respectively.

Returns

If succeeds: `FE_Success`

If fails: An error code

If API fails, the API also assigns one of the following values to FA_errno:

FA_errno value	Meaning
FE_Success	Operation is successful
FE_BadDocId	Invalid document ID
FE_BadDelete	objType is not FO_PgfFmt, FO_CharFmt, or FO_TblFmt

Changed APIs

F_ApiCompare ()

Compares two documents or two books.

Synopsis

#include "fapi.h"

. . .

F_CompareRetT F_ApiCompare(F_ObjHandleT olderId,

F_ObjHandleT newerId,

IntT flags,

StringT insertCondTag,

StringT deleteCondTag,

StringT replaceText,

`IntT compareThreshold);`

Arguments

New flag value added for the `flags` argument is as follows:

flags constant	Meaning
FF_CMP_ATTRIBUTES	Compare attributes for elements while comparing structured documents

Returns

New error codes returned are as follows:

FA_errno value	Meaning
FE_DocAlreadyHasTrackedEdits	The document being compared has tracked changes. Accept or Reject all changes before comparing.
FE_BookComponentAlreadyHasTrackedEdits	The book component being compared has tracked changes. See the console for a list of files. Accept or Reject all changes before comparing.

F_ApiSave

Saves a document or book. It allows you to script the way in which FrameMaker saves the file and to specify responses to warnings and messages that appear while the file is being saved. Save a file in its current name, or save it as a new file.

Synopsis

#include "fapi.h"

```
. . .
F_ObjHandleT F_ApiSave(F_ObjHandleT Id,
     StringT saveAsName,
     F_PropValsT *saveParamsp,
     F_PropValsT **saveReturnParamspp);
```

Returns

New error codes returned are as follows:

FA_errno value	Meaning
FE_FailedExportedFileInvalid	The setting Do NOT allow exporting of Invalid XML is enabled, and export operation is resulting in an invalid file. As a result, the save operation is canceled.

F_ApiSimpleImportFormats()

Imports formats from a document to a document or a book. If you import formats to a book, `F_ApiSimpleImportFormats()` imports formats to each book component for which the `FP_ImportFmtInclude` property is set to `True`.

Synopsis

```
#include "fapi.h"
. . .
IntT F_ApiSimpleImportFormats(F_ObjHandleT bookId,
F_ObjHandleT fromDocId,
IntT formatFlags);
```

Arguments

New value is added for formatFlags as follows

This value	To
FF_UFF_FBA	Import FBA expressions.
`FF_UFF_COND`	Import Boolean conditional expression as well with conditions.

F_ApiDialogEvent()

Is a callback that you can define in your client. The FrameMaker product calls `F_ApiDialogEvent()` when an event occurs in a dialog box opened by your client.

Synopsis

```
#include "fapi.h"
. . .
VoidT F_ApiDialogEvent(IntT dlgNum,
IntT itemNum,
IntT modifiers);
```

Arguments

These are the newly added negative constants that can be set for itemNum:

Constant	Meaning
FV_DlgInit	Dialog is initialized but not yet visible.
`FV_DlgHide`	The user hides the dialog
FV_DlgShow	The user unhides the dialog

F_ApiFind()

Performs the same actions as available in the Find dialog box to search a document for text or other types of content.

Synopsis

```
#include "fapi.h"
. . .
F_TextRangeT F_ApiFind(F_ObjHandleT docId,
const F_TextLocT *textLocp,
const F_PropValsT *findParamsp);
```

Arguments

Following are the newly added values for FS_FindObject:

Property	Value
FS_FindObject	FV_FindPgfFormatOverride
	FV_FindCharacterFormatOverride
	FV_FindTableFormatOverride

New 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_DialogResource properties

`FP_DockDialog`

`FP_IsDialogDocked`

`FP_IsDialogVisible`

FO_Doc properties

`FP_BkColor`

`FP_ReviewerNameList`

`FP_TrackChangesAddedColor`

`FP_TrackChangesDeletedColor`

`FP_UseInitialStructureOfAutoInsertedElements (For structured documents only)`

`FP_ShowElementDescriptiveNames (For Structured Documents Only)`

FO_Book properties

`FP_UseInitialStructureOfAutoInsertedElements (For structured books only)`

`FP_ShowElementDescriptiveNames (For structured books only)`

FO_Session properties

`FP_EnableAutoSpellCheck`

`FP_StructAppAttrConfigFile`

`FP_ProgId`

`FP_IsFMRunningAsServer`

`FP_AllowNewFileURL`

`FP_DoPostXSLTValidationOnExport`

`FP_DoNotExportInvalidXML`

FP_SuppressXMLParserWarnings

FP_RemoveExtraWhiteSpacesOnXMLImport

FP_NoMultiMediaInPDF

FO_CharFmt properties

FP_BkColor

FP_UseBkColor

FO_PgfFmt Properties

FP_BkColor

FO_Pgf properties

FP_BkColor

FO_FmtChangeList properties

FP_BkColor

FO_CondFmt properties

FP_BkColor

FP_UseBkColor

FO_Inset properties

FP_InsetPosterFile

FO_ElementDef properties

FP_AlsoInserts

FP_DescriptiveTag

FP_ElementDescription

FO_DlgButton properties

FP_icon

New property value constants

FP_DockDialog Values

- FV_DIALOG_DOCK_NONE
- FV_DIALOG_DOCK_LEFT
- FV_DIALOG_DOCK_RIGHT
- FV_DIALOG_DOCK_BOTTOM
- FV_DIALOG_DOCK_ALL

New script parameters

FS_DitavalCondTag

New notifications

FA_Note_Not_AI_Supported_File

FA_Note_AI_Supported_File

New FDE API

F_StrISubString()

Finds the first occurrence of a specified string in another string, ignoring case.

Synopsis

```
#include "fdetypes.h"
#include "fstrings.h"
. . .
IntT F_StrISubString(ConStringT s1,
ConStringT s2);
```

Arguments

s1	The string in which to search for `s2`
`s2`	The string to search for

Returns

The index of the first occurrence of `s2` in `s1`, or `-1` if `s2` doesn't occur in `s1`.

Adobe Systems Incorporated • 345 Park Avenue, San Jose, CA 95110-2704 USA • www.adobe.com