Adobe

FDK 7.0 Sample Clients

<Model Season="Summer" Id="BABCHEHF" ModelNumber="146 146 010">
<Name>XT-7 Uptown Style</Name>
<Picture
<Graphic xmlink="locator" href="images/UserGuidea.gif"
show="embed"
</Graphic>
window.onresize = NCSSFix;

Adobe FrameMaker 7.0 FDK Sample Clients

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA

Sample	Description	Win	Unix	Mac
`abc`	Demonstrates how to create and maintain a client owned inset. It works in combination with the `geoserver` sample client. Choose menu items provided with this client to insert the text insets, then use the additional menu choices from the `geoserver` client to make modifications to those text insets.	✓	✓	✓
`addstructrow`	Demonstrates how to add a row to an existing structured table.	✓	✓	✓
`aframes`	Generates a document that lists the page number, position, and size of all the anchored frames on body pages in the active document.	✓	✓	✓
`async`	Demonstrates asynchronous clients on Unix. This client can be compiled into two different sample clients to show different aspects of communication with a FrameMaker process. In one version, the sample client adds a menu to the document window menu bar and then waits for the end user to choose an item from the menu. In the second version, the sample client forcibly adds text to an open document. The client is described in detail in the Unix FDK Platform Guide. **NOTE:** To compile this client, you must make some edits to the Makefile. The required changes are detailed in comments within the Makefile.		✓	
`badlinks`	Generates a document that lists unresolved hypertext links. This document lists the page number and marker text of any hypertext marker with an unresolved link to a document, page, cross-reference, or graphic object. The client also writes errors to the console.	✓	✓	✓
`clickprt`	Brings up the print user interface for the filename that is passed as an argument in the hypertext message. Responds to a “message client”-style hypertext command.	✓	✓	
`cond`	Adds or removes the comment condition for the currently selected text.	✓	✓	✓
`ddeclnt`	Demonstrate registering FrameMaker client as a DDE server. This client registers as a DDE application the client named `FrameDdeSample`. The client accepts both `execute` and `request` transactions.	✓		
`dealloc`	Constructs and then displays a message in an alert box. The client builds a list of paragraph formats in the current document, determines the size of the list, then displays the number of paragraph formats in an alert box. This client demonstrates how to free resources allocated during string function calls.	✓	✓	✓
`dialog`	Creates a sample dialog box. A `.dre` file for the dialog box is included in the source. See the appropriate FDK Platform Guide for information about creating and compiling a dialog box resource from this file.	✓	✓	✓
`eewriter`	Converts upper case characters to lower case as the user types in a document.	✓	✓	✓

Sample	Description	Win	Unix	Mac
`elemutil`	Demonstrates how to manipulate elements and other FrameMaker structure objects. This client adds a menu called Element-Utils with the following commands: ● Format — Inserts the Emphasis element at the insertion point, or wraps the current selection in Emphasis. It then sets the Type attribute. ● Collapse All — Collapses the entire structure view. ● Expand All — Expands the entire structure view. Collapsing and expanding demonstrates how to traverse the structure tree. ● Reset Attributes — Resets all attributes to their default values. ● Add Acronym — Adds an `Acronym` and `Defn` element pair alphabetically to the `AcronymList` element. Demonstrates use of a custom dialog box and how to add an element. Source includes a `.dre` file for the dialog box. See the appropriate FDK Platform Guide for information about creating and compiling a dialog box resource from this file.	✓	✓	✓
`filter`	The sample filter client for the filter that is used in examples throughout the FDK documentation. This client filters text files into Frame product documents.	✓	✓	✓
`font families`	Displays a list of fonts that are available in the paragraph designer.	✓	✓	✓
`geoserver`	Works in conjunction with the `abc` client to demonstrate how to create and maintain a client owned inset.	✓	✓	✓
`getset`	Demonstrates how to get and set document and session properties. This client includes the following commands: ● BORDERS_CMD — Toggles Borders on and off ● ZOOM_CMD — Sets a document's zoom factor to values between 25% & 1600% ● HOST_CMD — Queries the current session for the current host name.	✓	✓	✓
`hello`	Greets the user at product startup time.	✓	✓	✓
`hello2`	Provides a menu selection to display the alert messages “Hello, World!” or “Goodbye, World!”.	✓	✓	✓
`hellowho`	Greets the user with a custom greeting established at product startup time.	✓	✓	✓
`hibye`	Demonstrates how to use notifications. Upon receipt of an `FA_Note_PostOpenDoc` notification, the client calls the function `sayHello()` to display an alert. Upon reception of an `FA_Note_PreQuitDoc` notification, the client calls the function `sayGoodby()` to display a different alert.	✓	✓	✓
`import`	Demonstrates how to import graphic files.	✓	✓	✓

Sample	Description	Win	Unix	Mac
`includelists`	Uses MIF statements or book component properties to determine which book components to print, or into which components to import formats or element definitions. This creates an effect similar to the user interface for these actions in versions of FrameMaker prior to FrameMaker 6.0. The client uses MIF statements or book component properties to determine its behavior as follows: ● File>Print Files in Include List — Prints every component in a book whose `DefautPrint` MIF statement is `Yes` (or whose `FP_PrintInclude` property is set). ● File>Import>Formats to Files in Include List — Imports formats into the book components whose `DefaultApply` MIF statement is `Yes` (or whose `FP_ImportFmtInclude` property is set). ● File>Import>Element Definitions to Files in Include List — Imports element definitions into the book components whose `DefaultApply` MIF statement is Yes (or whose `FP_ImportFmtInclude` property is set).	✓	✓	✓
`mainloop`	This directory contains sample clients that illustrate how to program asynchronous clients that run in main loops. **NOTE:** To compile these clients, you must make some edits to the Makefiles. The required changes are detailed in comments within each Makefile. The sample clients in this directory are: ● `doselect` A client that executes its own `select()` calls. The input file descriptor for this client is the the shell window from which you start FrameMaker. The client uses `F_ApiAlert()` statements to echo shell input in alert boxes. ● `noselect` A client that does *not* execute its own `select()` calls. The input file descriptor for this client is the the shell window from which you start FrameMaker. The client uses `F_ApiAlert()` statements to echo shell input in alert boxes. ● `xfmtest` A sample X Window System application. This client displays a modeless window with a menu. When the user chooses Quit from the menu, the client closes the active FrameMaker document (without saving changes) and exits.		✓	
`markers`	Demonstrates how to work with markers, and how to use menu separators. This client includes the following commands: ● INSERT_CMD — Inserts a marker of a specified type. The user is prompted for the name of the marker type. If the user specifies a string for a marker type, the client inserts a new marker of that type. The client also prompts the user for marker text. ● COUNT_CMD — Count all anchors within a selected text range.	✓	✓	✓

Sample	Description	Win	Unix	Mac
`mcount`	Counts markers in the active document, and reports the number back to the user.	✓	✓	✓
`mmlimport`	A filter client for opening and importing `mml` files.	✓	✓	✓
`myapi`	The example application presented in Chapter 1 of the FDK Programmer's Guide. It adds a menu to the document menu bar, with commands to: ● Close the current document ● Set the fill pattern of a selected object to black ● Add a page at the beginning of the current document.	✓	✓	✓
`newfmt`	Displays two dialog boxes to get user input, then uses the input to create a new paragraph format. The client then adds the new format to the document’s paragraph catalog.	✓	✓	✓
`openbook`	Performs a scripted open of all files in a FrameMaker book.	✓	✓	✓
`openxmldoc`	Demonstrates how to open an XML document and import it into FrameMaker.	✓	✓	✓
`otherapp`	A simple client that connects to a FrameMaker client which is registered as a DDE server. The client displays a small window for the user to connect to the FrameMaker client and send either `execute` or `request` transactions. The client interprets `execute` transactions as hypertext commands. For example, you can execute the following hypertext command syntax: `gotopage 3`. The client interprets specific `request` transactions. For example, a request of `CharTag` directs the client to retrieve the character tag of the beginning of the selection in the active document.	✓		
`PageNo2`	Demonstrates a way to number pages in a document separately from the numbering properties that are set up for a book.	✓	✓	✓
`pcount`	Counts the pages in all components of a FrameMaker book.	✓	✓	✓
`pcount2`	Counts the pages in a FrameMaker document.	✓	✓	✓
`pickfmts`	Allows the user to choose a group of paragraph formats to delete from the active document. This client uses a custom dialog box.	✓	✓	✓
`reverse`	The Reverse Translator reverses the order of two elements upon translation from SGML to structured FrameMaker, and vice versa. Upon export, it takes a FrameMaker document which contains the element `First` followed by the element `Last` and saves it as an SGML document with `Last` followed by `First`. This client illustrates the use of `Sw_ScanElem()` to change the order of elements on export. **NOTE:** To compile this client on Unix platforms, you must make some edits to the Makefile. The required changes are detailed in comments within the Makefile.	✓	✓	✓

Sample	Description	Win	Unix	Mac
`saveas`	Saves a document as PDF or HTML, depending on the menu choice. This client also demonstrates how to set the hiearrchical level of PDF bookmarks via the FDK.	✓	✓	✓
`setpdfdocinfo`	Demonstrates how to set object properties for Tagged PDF.	✓	✓	✓
`setxmpinfo`	Demonstrates how to set XMP info through the API	✓	✓	✓
`sgmlplus`	This is a C++ development environment for writing Structure Import/Export API clients. Note that `sgmlplus` is written on top of the Structure Import/Export API—it is not a distinct C++ API. For information about this environment, see the description in the included file, `sgmlplus.fm`. **NOTE:** To compile this client on Unix platforms, you must make some edits to the Makefile. The required changes are detailed in comments within the Makefile.	✓	✓	✓
`struct`	This sample client lays a basic foundation for all SGML or XML import/export clients, no matter how complex they become. **NOTE:** To compile this client on Unix platforms, you must make some edits to the Makefile. The required changes are detailed in comments within the Makefile.	✓	✓	✓
`sumcols`	Sums the columns in a table. This client sums only integer values. To obtain a number it looks for digits at the start of a table cell. The sum is placed in a footer row at the end of the table. Values in header rows are ignored.	✓	✓	✓
`tables`	Demonstrates how to traverse and manipulate tables, rows, cells, and cell text. This client includes the following commands: ● ITEM1—Deletes text from the cell that contains the current selection. ● ITEM2—Gets the row and column position of a single selected cell. ● ITEM3—Gets the integer value representing the position of the last Body Row. ● ITEM4—Gets the Integer value representing the position of the first Body Row. ● ITEM5—Determines whether a cell is set to Straddled or not. ● ITEM6—Places the cursor at the end of the paragraph (within a cell) that it is currently inserted in.	✓	✓	✓
`tag`	For any text range that has a format override of italic, this client changes that range to use the emphasis character tag.	✓	✓	✓
`text`	Demonstrates how to manipulate text. This client gets text from the document and prints it to console, adds text to the document, deletes text, changes text point size, and changes text color.	✓	✓	✓

Sample	Description	Win	Unix	Mac
`uservar`	Writes a user name into a variable named “User Name”, when an `FA_Note_PostOpenDoc` notification is received. User name is acquired via the session property `FP_UserName`.	✓	✓	✓
`version`	Gets and displays the major/minor version number of FrameMaker. Makes use of string functions. This client makes calls to the functions `getProductName()`, `getVersionMajor()`, and `getVersionMinor()`. For more information about these functions, see the included file `cutils.c`	✓	✓	✓
`viewonly`	Sets all open documents to view only, and adds some text to each document at its last insertion point location.	✓	✓	✓
`whatpage`	Identifies the Body page number of the selected object. This client demonstrates how to find the body page that contains the current selection for selected text or graphic objects.	✓	✓	✓
`wordcnt`	Counts words in the current document. It locates all flows on body pages in the document, and reads the content of every paragraph in each flow. It then counts the words in each paragraph. A character is defined to be either an “in-word” or a “not-in-word” character. A “word” is a group of one or more “in-word” characters—delimited on either side of the group by a “not-in-word” character, a paragraph break, or a hard (user-inserted) line break. In addition the program counts words in graphic text lines which appear on body pages of the document. Text in footnotes, tables, or generated from variables or cross references is counted. Neither text generated by autonumber strings nor text on master or reference pages (e.g. headers and footers) is counted.	✓	✓	✓