

CudSpan Plugin Documentation



CudSpan

COPYRIGHT NOTICE AND DISCLAIMER

BookLooker, DoBatch, CudLog, TocBreaker, PGFWhopper, RefIcons, PrintTree, and HuntOverrides, as mentioned in this document, and irrespective of capitalization, are CudSpan FreeWare. You may freely use and distribute these plugins as long as you do not modify them in any way.

The CudSpan plugins mentioned here are provided AS IS. The author of these programs accepts no responsibility in any way for what may happen as a result of your use and distribution of these programs.

This documentation is provided AS IS — The author accepts no responsibility in any way for what may happen as a result of your use of this documentation.

CudSpan is a capricious name that serves as a pseudonym for Chris Despopoulos; all rights reserved.

All product and company names mentioned in this document are the registered names and/or trademarks of their respective companies.

Copyright ©, 2001, CudSpan (Chris Despopoulos), all rights reserved.

CudSpan Table of Contents

CudSpan	
Introduction	
Freeware by CudSpan	1
CudSpan	
BookLooker & HuntOverrides Plugins	
Using BookLooker and HuntOverrides.....	3
Calling HuntOverrides from DoBatch	4
CudSpan	
PgfWhopper Plugin	
Using PGFWhopper	5
CudSpan	
PrintTree Plugin	
Using PrintTree	7
Calling PrintTree from DoBatch	8
CudSpan	
Reflcons Plugin	
Installing the DLL and the Palette Document:.....	9
Configuration.....	10
Using REF_ICONS	10
Displaying and Closing the Palette	10
Inserting Icon/Format Pairs	10
Adding an Icon/Format Pair to the Palette	11
Deleting an Icon/Format Pair from the Palette	11
Modifying the Palette Document	12
Specifying the icon/format pair	12
Adding the pair to the popup menu	12
Installing and testing the modified palette document	13
CudSpan	
MarkerWorker Plugin	
Overview	15
Using MarkerWorker	16
CudSpan	
DoBatch Plugin	
Installing the DLL Files.....	17
Running a Batch.....	18
Writing a Batch File.....	18
Generating a Batch Template	18
Getting DoBatch Help	18
Batch Commands.....	19
Using CudSpanToolBox.dll	22
General Syntax	22
Calling PDFSize	22
Developing Toolbox Plugins.....	23
Requesting Notification in your Client	23
Posting Log Messages from your Client	24
CudSpan	
TocBreaker Plugin	
Scenario for use	25

Getting Started	25
Using TocBreaker.....	25
Storing Page Break Information	26
Storing Line Break Information	26
About Applying Page and Line Breaks	27



CudSpan

CudSpan Introduction

1 Freeware by CudSpan

Welcome to the CudSpan set of FreeWare FrameMaker plugins. CudSpan is a whimsical name I use to separate issues related to my freeware from issues related to my gainful work. My work includes FDK and Structure Import/Export API programming, training, and general technical writing for the software and hardware industry. If you need a technical writer or somebody to modify FrameMaker, feel free to contact me at cud@telecable.es.

Document statement

The CudSpan set of tools is absolutely free. You do not need to pay me for the use of these plugins. But I ask you to consider the following:

DANGER: The world is in trouble.

The next time you feel like buying a useless piece of garbage,

JUST SAY NO!

You may use these plugins in any way, and you may distribute them to whomever you want. You may not modify these plugins. Also, you may not hold me responsible in any way for anything that may result because you use or distribute these plugins.

I hope you find these plugins useful. They are:

- **BookLooker:** a tool that performs various book-wide functions. It sets change bars, locks and unlocks files, and it searches for strings and markers in a book. Note the you must have FrameMaker version 5.5.6 or later.
- **CudLog:** a tool that posts messages to a log file. This is used by other plugins in the CudSpan set. If you are a developer and you want to post messages to CudLog from your own plugins, see the instructions for DoBatch.
- **DoBatch:** a tool that imitates fmbatch functionality for Windows versions of FrameMaker.
- **HuntOverrides:** a tool that lists paragraph format overrides in your documents or books.

- PgfWhopper: a tool that lists unused pgf formats, char formats, and condition tags for a given document. You can use it to delete any of these unused formats.
- PrintTree: a tool that prints out the structure tree of a FrameMaker+SGML document. You must have CudLog installed to use this plugin.
- RefIcons: a palette that inserts icons at the head of a paragraph and changes the paragraph format. This is good for notes, tips, and warnings, for example. Installation seems a little involved, but once you install it, I believe it is easy to modify and use.
- TocBreaker: commands to “remember” and apply page breaks and line breaks you inserted in generated files. If you ever had to generate a book 15 times in the last week of a deadline, you probably wished for this... I know I did, so I made it.



CudSpan

CudSpan BookLooker & HuntOverrides Plugins

BookLooker is a Win plugin that performs the following for all the files in a book:

- Searches for strings, xrefs, unresolved xrefs, markers of type, and marker text
- Locks and unlocks all files in a book
- Clears, turns on, and turns off changebars for all files in a book

Included in this set of plugins is HuntOverrides. HuntOverrides makes a report of all paragraph format overrides in a document or book.

1 Using BookLooker and HuntOverrides

The BookLooker search facility and HuntOverrides each generates a hypertext document that lists all the instances you searched for. You can click an entry in the hypertext document to go to any specific instance. HuntOverrides lists all instances of paragraph format overrides. It also shows when an override is caused by only a page or column break — in that case, the entry in the results document begins with “Page-break:”.

To install the plugins, just put the dlls into your PLUGINS directory. No need to tweak your ini file.

To use the BookLooker commands, a book window must be active. The commands appear in the CudSpan Book Tools menu. The BookLooker submenu lists the things you can search for in a book.

When HuntOverrides is installed, Hunt Overrides For Current Book and Hunt Overrides For Document both appear in the CudSpan Book Tools menu. Hunting for a book lists all overrides in the currently active book, and hunting for the document lists overrides in the currently selected book component.

When a document is active, Hunt Overrides for Document appears in the CudSpan Doc Tools menu. That command lists overrides in the current document.

You can also invoke HuntOverrides from a DoBatch script. For more information, see *CudSpan DoBatch Plugin*.

2 **Calling HuntOverrides from DoBatch**

You can call this plugin from a DoBatch script via the following:

```
callClient.huntOverrides(filename)
```

For example:

```
callClient.huntOverrides(C:\docbook\dbguide\fm\preface1.fm)
```



CudSpan

CudSpan PgfWhopper Plugin

PgfWhopper is a Win plugin that checks your document for any paragraph formats that are defined in the catalog but not used in the document. You then have the option to delete any of the listed format definitions from the catalog.

1 Using PGFWhopper

- To install the plugin, just put the dll into your PLUGINS directory. No need to tweak your ini file.
- The command is File > Utilities > PGF_WHOPPER > Check for Un-used Paragraph Formats. (I offer no apologies for the lame command name.) There are also commands to check for unused character formats and unused condition tags.
- When you choose the command, a dialog box appears with two scroll lists. The left one shows all the format definitions you have not used in the document. The right one shows all the format definitions you want to delete.
- To move an item from one list to the other, just click on the item.
- To delete the format definitions in the right-hand list, click OK.
- Don't worry, an alert appears before PgfWhopper whops your document, so you have a chance to cancel. And you can always revert. Unfortunately, there is no UNDO.

That's it... Pretty simple, eh?



CudSpan

CudSpan PrintTree Plugin

PrintTree is a Win plugin that generates a text representation of the structure tree of a FrameMaker+SGML document, as follows:

```
-Chapter ->
|
| -Title -> Block-Oriented Elements
|
| -Para -> Most book components can conta
|
| -ItemizedList
||
|| -ListItem
|||
||| -Para -> Highlights (list of main point
|||
||| -ListItem
|||
||| -Para -> an Epigraph
|||
||| -ListItem
|||
||| -Para -> Paragraphs
```

The output shows the hierarchy of elements. For any element that has text content, it also shows a snippet of that content. Notice that it is best viewed with non-proportional fonts. However, PrintTree writes the output to the CudLog log file — that file uses proportional fonts. You should copy the output and paste it into a document that you can modify to make the results as readable as possible.

PrintTree shows the tree as it appears in the Structure View — it does not show the children of any collapsed elements.

1 Using PrintTree

- To install the plugin, just put the dll into your PLUGINS directory. No need to tweak your ini file.
- You must also have the CudLog dll properly installed

- The command is CudSpan Doc Tools > Print Struct Tree for Document. There is a similar command for books.
- When you choose the command, PrintTree writes out the structure tree to the CudLog log file.

That's it... Pretty simple, eh?

2 Calling PrintTree from DoBatch

You can call this plugin from a DoBatch script via the following:

```
callClient.treePrint(filename)
```

For example:

```
callClient.treePrint(C:\docbook\dbguide\fm\preface1.fm)
```



CudSpan

CudSpan RefIcons Plugin

RefIcons is a Win plugin that automates the creation of icon/pgf format pairs, such as the following:



NOTE: Blah blah.

Notice, there is an icon, the text is indented, and there are lines above and below the paragraph. You use the feature by opening a palette and choosing the icon/pgf format pair you want to insert. The palette itself contains definitions for each icon/pgf format pair on its reference pages. This means that you can add or change the pair definitions by modifying the palette document... No need for programming or modifying the DLL.

If you can define a paragraph with an icon similar to the above example with the Note icon, you can define your own pairs. To use RefIcons, you need to understand the following:

- Installation - where to put the DLL and the palette document
- Configuration - you must modify the .ini file to tell Maker to keep the palette on top
- Use - the easiest part

1 Installing the DLL and the Palette Document:

To install the DLL, put it in your Plugins directory.

To install the palette document, you need to put it in the correct directory. The palette path must be

```
PRODUCT_DIRECTORY\fminit\ontop\IconsPalette.fm
```

where `PRODUCT_DIRECTORY` is the directory where you installed Maker, and `IconsPalette.fm` is the name of the palette document. For example, on my system, I have the palette installed in:

```
D:\Program Files\Adobe\Maker+SGML\fminit\ontop\IconsPalette.fm
```

To install the palette document:

1. Create a directory named `ontop` in the `fminit` directory for the version of Maker you will use.
2. Put a *copy* of the `IconsPalette` file into that directory.
3. Store the original version of the palette document in a safe place.

2 Configuration

Once you have the palette document installed, you need to edit your `.ini` file. The `.ini` file has entries to specify a directory for general palettes, and a directory for palettes that always stay on top of your other windows. By default, no directory is specified for palettes to stay on top. You will specify a directory named `ontop` for this purpose.

- In your `.ini` file, search for `AlwaysOnTopPaletteDir`.
- Change this entry to `AlwaysOnTopPaletteDir=fminit\ontop`.
- Save your `.ini` file.

3 Using REF_ICONS

When `REF_ICONS` is properly installed and configured, there are four things you can do with this tool:

- Display and close the palette
- Insert an icon/format pair in your document
- Add an icon/format pair definition to the palette
- Delete an icon/format pair from the palette

3.1 Displaying and Closing the Palette

To display the palette, choose `Special > Display Icons Palette...` To close it, click the close box.

3.2 Inserting Icon/Format Pairs

With the palette displayed:

1. Put the insertion point in a paragraph or make a selection in a paragraph.
2. Click `Choose Icons` in the palette, and choose an icon/format pair from the popup menu. The selected paragraph changes to show the associated icon, and the properties of the associated paragraph format.

WARNING: If the paragraph you select already has an icon, you should delete that icon before applying a new icon/format pair. REF_ICONS does not delete existing anchored frames from a paragraph.

3. If you do not have the new format defined in your document's format catalog, the new format will appear as an override. You may want to apply the new format to your catalog.

NOTE: Assume the icon/format pair is defined in the palette to include a frame above or below the paragraph. When you insert the format pair in your document, the frames may not appear in your document. This indicates that the document you are editing does not have the same frames set up on the Reference reference page. (These frames are set up to display lines above and below paragraphs.) To remedy the situation, you only need to define reference frames with the appropriate names. Go to the document that was the original source of this icon/format pair, and look on the reference page for the necessary frame names.

3.3 Adding an Icon/Format Pair to the Palette

The palette includes an Add Icon button to add a pair to your palette.

NOTE: The palette document contains all the definitions for the icon/format pairs. You can use the palette to insert these pairs in any FrameMaker document. This is true whether or not your document has a corresponding paragraph format in its catalog. If your document does not have a corresponding paragraph format in its catalog, the result will be a paragraph in your document that appears with format overrides. You may want to ensure your templates have the same paragraph formats as those defined in the icon/format pairs of the palette document.

1. Create and name a paragraph format that accommodates the dimensions of your icon. Then insert an anchored frame in the paragraph at the first character position in the paragraph. It is important that the frame anchor is the first character in the paragraph.
2. Put the IP in that paragraph, or make a selection in the paragraph.
3. In the palette, click Add Icon.
4. In the dialog box that appears, provide a name, and click OK.

3.4 Deleting an Icon/Format Pair from the Palette

The palette also includes a Delete! button to delete a pair from the palette.

1. In the palette, click Delete!
2. Type the name of the icon/format pair you want to delete, then click OK.

4 Modifying the Palette Document

In case you're curious, the following describes what happens when you add and delete format pairs in the palette. Actually, the first version of this dll didn't have the Add Icon and Delete! buttons on the palette. To define an icon/format pair, you had to do the work manually. So, to explain what happens behind the scenes, I'm just including the old instructions for modifying the palette.

WARNING: The following instructions are obsolete. Kids, don't do this at home.

NOTE: When you installed the palette doc, you put a *copy* of it in the `ontop` directory. So you should have the original stored in a safe place. If you open the palette document from a location other than the `ontop` directory, you can see what it contains. First open the file, then unlock it by typing `ESC-F, l, k`.

4.1 Specifying the icon/format pair

This is the first step... You will define a pair on a reference page, and then store the pgf format in the catalog of the palette doc.

1. View Reference Pages.
2. Create a new Reference Page - You can give any name you like to the Reference Page.
3. Draw a text frame on the Reference Page.
4. Name the text frame - This name should relate to the purpose of the icon/format pair. You will use this name to pass your pair to the DLL via the popup menu.
5. In the text frame, apply a paragraph format to its first paragraph.
6. Put the IP to the left of the first character in the paragraph.
7. Insert an anchored frame and put your icon into it.
8. Format the anchored frame as you like.
9. Save the document.

4.2 Adding the pair to the popup menu

Once you have a new icon/format pair, you need to add it to the popup menu. This menu uses message hypertext links to pass the appropriate information to the DLL.

1. With Reference Pages showing, go to the page named `Commands`.
2. Create a new entry in the list that appears on that page. It is probably easiest to just copy one entry and paste it at the end of the list.
3. Change the text of the entry to give it the command name you want. The name of the icon/format pair is usually a good choice.
4. Open the Markers dialog box.
5. Triple-click the new entry to select the entire paragraph. If you copied an existing entry to create this one, you should see text in the Marker Text box, and the Marker Type should be Hypertext.

6. Enter the appropriate marker text for your new entry. If you copied an entry and it has a proper marker in it, then you can simply edit the marker text. If there is not already a marker, then type in the text, and make sure the Marker Type is Hypertext.

The syntax for your marker text must be:

`message ref_icons IconsPalette:TextFrameName`, where `TextFrameName` is the name of the text frame that contains your new icon/format pair. For example, if you created a new pair for examples, and defined it in a text frame named `EXAMPLE`, you would type `message ref_icons IconsPalette:EXAMPLE`.

7. In the Marker dialog box, click New Marker.
8. Save the document

4.3 Installing and testing the modified palette document

Once you have changed the palette document, you simply copy the new version to the `ontop` directory. Of course, you should save a copy of the modified palette in a safe place.

1. Through the operating system, copy the palette document and place the copy in the `PRODUCT_DIRECTORY\fminit\ontop\` directory, where `PRODUCT_DIRECTORY` is the directory where you installed Maker.
2. In Maker, if the palette is already open, close it.
3. Open the palette through the menu command, then use your new command and check your results. You should also check all the other commands to ensure you have not introduced any bugs.



CudSpan

CudSpan MarkerWorker plug-in

MarkerWorker is a Win plug-in that generates a list to represent all markers of a specified type in a book or document. Once you generate the list, you can modify the marker text and then rebuild all the listed markers with your new marker text. Obvious uses include:

- Globally editing index markers
- Translating marker content
- Checking and correcting hypertext markers

1 Overview

To use the plug-in, you first generate a list of markers of a given type. The list appears as follows:

MarkerWorker List of Hypertext Markers

```
****  
openObjectId Intro.fm:2 256706  
****  
openObjectId Intro.fm:2 606259  
****  
openObjectId BookLooker.fm:2 606503  
****  
openObjectId BookLooker.fm:2 606525  
****  
openObjectId BookLooker.fm:2 606577
```

The output in a marker list is a pair of paragraphs for each marker — the first paragraph is “****”, and the second is the text of the marker.

The first paragraph of the output pair includes an index marker that contains information to identify the marker you will modify. (The language for discussing a marker that identifies a marker is a bit baroque — please bear with me. It’s not as complicated as the language would imply.) For example, the first “****” paragraph in the above list includes an index marker with the following content:

```
C:\Docs\CudSpanDocs\TOC.fm%580896%openObjectId Intro.fm:2 256706
```

The marker text is a data string that identifies the file that contains the marker, the unique ID for that marker, and then the content of the target marker.

CAUTION: The data string is separated into fields by the “%” character — if your markers include that character, then MarkerWorker will fail to reliably identify markers to rebuild. If you believe some markers contain this character, you should search for those entries in the generated list *before* you edit it. Remove the marker pair for such an entry and edit that marker by hand.

If this is a problem please let me know.

The idea is that you edit the content of the second paragraph in the output pair. When you have edited all the marker text, you then choose a command to rebuild the markers with your edited text. As usual, before rebuilding your markers, you should save off a copy of your document or book. It is common sense to follow this practice before running any batch process.

WARNING: Do not delete the marker in the “*****” paragraph, or edit the paragraph in any way. Do not add any paragraphs to the list. If you delete paragraphs, be sure to delete the *complete output pair* of paragraphs. MarkerWorker relies on the “*****” paragraphs and on the specific order of output pairs in the generated list.

2 Using MarkerWorker

- To install the plug-in, just put the dll into your PLUGINS directory. No need to tweak your ini file.
- The command to build a list of markers for a document is CudSpan Doc Tools > Extract Specified Marker Text. There is a similar command for books. Choose the command to generate a marker list.
- In the dialog box that appears, type the type of marker you want to extract.
- Edit the paragraphs of marker text that appear in the generated list.
- To rebuild a set of markers, make the list document active, then choose CudSpan Doc Tools > Rebuild Markers From List.

CAUTION: Before rebuilding your markers, be sure to save a copy of your book or document. This is just common sense, and you should save a copy of your data before you perform *any* batch operation.

That’s it... Have fun!



CudSpan

CudSpan DoBatch Plugin

DoBatch is a Win plugin that is a system to run batches on your books or documents. CudSpan DoBatch is a plugin that does very few things. The batch plugin:

- Opens, saves, and closes files
- Saves files as FrameMaker documents, View Only, MIF, text, Post Script, or filtered
- Imports formats from one open document into another
- Updates cross-references and variables in a document — performs simple update/generate on a book
- Writes messages to a log
- Executes other plugins

This last action, executing other plugins, is unique. The batcher can execute any plugin that registers the correct notification, and that operates on an entire document or book.

There are two advantages to this approach; maintenance, and adding new functionality. To add new commands to your batch toolbox, or fix existing ones, you only need a developer to create or update a dll, and then you add it to your tool box. There are simple rules about what the dll needs in order to support the CudSpan batch scheme. Developers should see *Developing Toolbox Plugins* on page 23.

1 Installing the DLL Files

To use the CudSpan Batch system, you need the following plugin files;

- doBatch.dll
- cudLog.dll (optional but highly recommended — used for logging)
- CudSpanToolBox.dll (optional — includes various utilities you can call from a batch file)

To install them, you can put the dll files in any location within your `Plugins` directory. To keep the batcher and the tools together, you should make a directory inside your `Plugins` directory and name it `CudSpan`. Then put all your CudSpan dll files into that directory.

2 Running a Batch

NOTE: Any time you run a batch, you should save off a copy of your files. That's just insurance, in case you have the batch do a million terrible things in all your files.

Running a batch involves the following steps:

- Write a batch file
- Save a copy of your source files
- Choose Run Batch from the CudSpan Doc Tools menu
- Choose the batch file you want to run
- Look over the log for indications of problems
- Look over your results

3 Writing a Batch File

Included in your package is a file named `batch.fm`. This is an example batch you can use to help write your own batch files. You can use it as is, except you must change the filenames to match FrameMaker document files on your system.

A batch file must be a FrameMaker document. Each batch command is one paragraph — that's true whether the paragraph wraps or not. However, you can only have 255 characters in a command paragraph.

If the first character in the paragraph is '#', then the paragraph is a comment. It must be the *first* character... Otherwise it appears as an illegal command. You cannot have a command followed by a comment in the same paragraph — a paragraph is either a command or a comment. A comment can be as many characters as you like, so long as it is in one paragraph.

Blank paragraphs are ignored.

3.1 Generating a Batch Template

If you want to process all the documents in a book, you can begin with a batch template that includes commands to open and close every file in the book. You can use this as a basis for the batch you will run on the files.

To generate a batch template, make a book window active. Then choose CudSpan Book Tools > Build a Batch Template for Current Book.

3.2 Getting DoBatch Help

To see a list of commands, and their syntax, choose CudSpan Doc Tools > DoBatch Help. DoBatch writes out a listing of commands to the log. Note: because DoBatch is freeware, you cannot rely on CudSpan to provide comprehensive technical support.

4 Batch Commands

Command syntax is as follows: `command.subcommand(arg)`. Following are the commands supported by the CudSpan DoBatch plugin:

log

```
log.message(string)
```

If a log file already exists, writes a message to the log. If no log file exists, opens a log file and writes the message to it.

NOTE: For this command to work, the CudLog plugin must be installed.

open

```
open.book(path)
open.book.visible(path)
open.doc(path)
open.doc.visible(path)
```

These commands open a document or book file. `Invisible` is the default; add the `.visible` subcommand to open the file and display it.

NOTE: If you open a document invisibly, be sure to close it via DoBatch. If you fail to close the document, it will still be locked the next time FrameMaker tries to access the document. For example, if you open a document invisibly, then quit FrameMaker, that document will still be locked the next time you start FrameMaker.

save

```
save.book(path)
save.doc(path)
```

Saves a document or book. You must save a file before you close it if you want to save any changes.

saveAs

```
saveAs.book.fasl(path path)
saveAs.doc.fasl(path path)
saveAs.book.mif(path path)
saveAs.doc.mif(path path)
saveAs.book.text(path path)
saveAs.doc.text(path path)
saveAs.doc.ps(path path)
saveAs.doc.filter(path path filterHint)
```

Save As for a document or book. The first path is the path of the file you want to save, and the second path is the path and filename you want to save it to.

NOTE: After you save an open document as fast to a different name, the file with the older name is no longer open — the batch can no longer identify that file by its older pathname. Subsequent batch commands must refer to the new filename.

When saving as PostScript, DoBatch actually uses the FrameMaker command to print to a file. Before running a batch to save as PostScript, be sure to choose the appropriate printer (if you will create PDF, you should use Acrobat Distiller as the printer), and make any appropriate setting. You make these choices and settings via the File > Print Setup command.

When saving filtered files, `filterHint` is a string that specifies which filter to run (what filtered format you want to save to). Filter hint strings can include spaces, so they should always be specified in quotes. The following table lists some common filters and their hint string. This list does not indicate any preference for software programs or file formats.

Filter Hint String	Resulting File Format
"0001AW4W0192"	Microsoft RTF
"0001AW4W0490"	Microsoft Word Win 6.0/7.0
"0001AW4W049m"	Microsoft Word Mac 6.0
"0001AW4W0542"	Microsoft Word Mac 5.x
"0001AW4W0541"	Microsoft Word Mac 4.x
"0001AW4W0071"	WordPerfect DOS/Win 5.1
"0001AW4W0602"	WordPerfect Mac 3.5
"0001IVY RTFJ"	RTF (Japanese) — Note the space
"0001ADBEHTML"	HTML
"0001ADBEXML "	XML — Note the trailing space

For more information about these hint strings, look for *Import File Hint Strings* in the Adobe FDK documentation.

close

```
close.book(path)
close.doc(path)
```

Closes a document or book without saving the file.

importFormats

```
importFormats.apflcvrtxkmdBO(path1 path2)
```

Imports formats from `path2` into `path1`. The characters you provide as the subcommand determine which formats to import, as follows:

- a - import all formats and definitions - does not retain overrides.
- p - pgf formats
- f - char formats
- l - page layouts
- c - xref formats
- v - variable formats
- r - ref pages
- t - table formats
- x - cond text settings
- k - color definitions
- m - math definitions
- d - doc properties
- B - Retain manual page breaks
- O - (upper case "O") - Retain other format overrides

NOTE: If you specify `a` in the subcommand, it overrides all other flags except `B` and `O`. `DoBatch` ignores unknown characters that appear in this subcommand.

update

```
update.book(path)
update.doc(path)
```

For a document, updates the variables and cross-references in the document. For a book, performs a simple Update/Generate.

callClient

```
callClient.[clientName](string)
```

Executes the specified plugin. You provide the plugin name as a subcommand, and you specify the plugin arguments in `string`.

5 Using CudSpanToolBox.dll

CudSpanToolBox.dll is a special plugin that I developed to run various plugins that ship with the FrameMaker product. You call the toolbox dll from a batch command, and that dll calls the FrameMaker dll to execute a process on a specified document or book.

As of this writing, the toolbox only supports the PDFSize plugin. This plugin ships with FrameMaker 6.0 — it serves to bring earlier version FrameMaker documents up to date with the 6.0 method of reducing PDF file size. PDFSize automates the Format > Document > Optimize PDF Size command.

5.1 General Syntax

To use the toolbox dll in a batch file, type a command with the following syntax:

```
callClient.cudSpanToolBox(dllName@string@string...)
```

`dllName` is the name of the dll you want to call, and the following strings are the arguments to pass to that dll. Note that each string is separated by the @ character.

5.2 Calling PDFSize

The easiest way to call PDFSize is to specify a file to process. You can set the optimization options in FrameMaker via the Format > Document > Optimize PDF Size > Options command. Once you set these options, they remain in effect until you change them again. After you set your options, run a batch that opens your files, calls PDFSize for each file, then saves the files.

To run PFSize on a specific document file, specify the keyword `File`, and then the filepath for that document. For example, the following commands open, optimize, save, then close a document named `myFile.fm`:

```
open.doc.visible(C:\myFile.fm)
callClient.cudSpanToolBox(PDFSize@File@C:\myFile.fm)
save.doc(C:\myFile.fm)
close.doc(C:\myFile.fm)
```

You can also specify a script file that defines the list of documents to process, and the optimization settings to use. For information on the script file, see the FDK Reference Manual. To call PDFSize with a script, specify the keyword `Script`, and then the filepath for the script. For example, the following command calls PDFSize to run with a script named `myScript.txt`:

```
callClient.cudSpanToolBox(PDFSize@File@C:\myScript.txt)
```

6 Developing Toolbox Plugins

NOTE: The following is not intended to be documentation for FDK programming, nor instructions on registering clients to receive `F_ApiCallClient()` calls.

You *must* consult the Adobe FDK documentation for reliable FDK programming instructions.

DoBatch can call another plugin as one of its commands. However, to work with DoBatch, a plugin must follow these rules:

- The plugin must be designed to operate on an entire document or book — DoBatch commands can only pass a filename to your plugin to identify what it is your plugin will operate on
- The plugin must be able to convert a pathname into a document or book ID
- The plugin must be registered with a name that has no spaces
- The plugin must request `CallClient` notification
- To post log messages, the plugin must include the `writeLog()` function

In DoBatch, a `callClient` command is the same as executing `F_ApiCallClient()` in the FDK. For example,

```
callClient.huntOverrides(c:\mydoc)
```

in DoBatch is the same as

```
F_ApiCallClient("huntOverrides", "c:\\myDoc");
```

in an FDK client.

6.1 Requesting Notification in your Client

To receive the call from DoBatch, your client must request the proper notification, as follows:

```
F_ApiNotification(FA_Note_ClientCall, True);
```

The client must also include a notification handler, as follows:

```
VoidT F_ApiNotify(notification, docId, sparm, iparm)
IntT notification;
F_ObjHandleT docId;
StringT sparm;
IntT iparm;
{
}
```

In this handler, `notification` identifies the notification call (in this case, `FA_NOTE_ClientCall`), and `sparm` contains the string argument DoBatch is able to pass. In the above example, `sparm` would contain the pathname `"c:\myDoc"`.

Then your handler must include the code to parse the string in `sparm`, and perform the desired actions.

6.2 Posting Log Messages from your Client

DoBatch includes a logging utility dll named CudLog. You can call that dll to post messages to the DoBatch log file — if the user has CudLog installed, your messages will appear. If it is not installed, your call to post the message returns an error, and is otherwise ignored by the FrameMaker session.

To post log messages to CudLog, include the following function in your client:

```
IntT writeLog(StringT s)
{
    StringT tmpStr;

    tmpStr = F_StrNew((UIntT)255+1);
    F_StrCatN(tmpStr, (StringT)"logMessage@", (UIntT)255);
    F_StrCatN(tmpStr, s, (UIntT)255);
    F_ApiCallClient((StringT)"cudLog", tmpStr);
    if(!F_StrIsEmpty(tmpStr)) F_ApiDeallocateString(&tmpStr);

    return(0);
}
```

You call this function within your code as follows:

```
writeLog((StringT)"This is a log message");
```

NOTE: The string you pass to writeLog() should be no longer than
255 - F_StrLen("logMessage@");



CudSpan

CudSpan TocBreaker Plugin

TocBreaker is a Windows plugin that is good for the later phases of book production.

1 Scenario for use

Imagine you have a book that is past the Beta stage. The TOC is very stable, and so is the index. However, you find yourself generating the book because you are apt to make changes that move sections or index sources to different pages. Anyway, you should regenerate, just in case you performed some actions that can invalidate the Auto Hypertext in your generated files.

However, you find that you had to apply page breaks and line breaks to some of the entries in your generated files. Every time you regenerate the book, those changes are lost. What you want is a command that tells FrameMaker to “remember” the page breaks and line breaks. TocBreaker gives you commands to do that, and to apply your breaks when you want to.

2 Getting Started

- To install the plugin, just put the DLL into your PLUGINS directory. No need to tweak your .ini file.
- The commands are under File > Utilities > TOC Breaker Commands. (I offer no apologies for lame command names.) They are divided into two categories - Apply Page/Line Breaks and Store Page/Line Break Information.
- When you store the information, TocBreaker creates a reference page (if it isn't already there) and writes data to that page.
- When you apply breaks, TocBreaker loops through the data on the reference page, finds the associated paragraph in the main flow, and applies the page break or inserts the hard line break.

3 Using TocBreaker

The following sections describe each step to use TocBreaker in detail.

3.1 Storing Page Break Information

With your generated file active, choose Utilities > TOC Breaker Commands > Store Page Break Info. This command loops through the main flow of your document, counting paragraphs as it goes, and storing information about paragraphs with page breaks. Specifically, it:

- Creates a TocBreakerPage reference page, if needed
- Creates 4 flows, one for each type of page break, if needed
- If a previous set of data exists, deletes the existing data
- For a paragraph with a page break, writes the paragraph count to the appropriate flow
- For a paragraph with a page break, writes the paragraph text (255 chars, max) to the appropriate flow

3.2 Storing Line Break Information

With your generated file active, choose Utilities > TOC Breaker Commands > Store Line Break Info. This command loops through the main flow of your document, counting paragraphs as it goes, and storing information about paragraphs with hard line breaks. Specifically, it:

- Creates a TocBreakerPage reference page, if needed
- Creates 1 flow for line break info, if needed
- If a previous set of data exists, deletes the existing data
- For a paragraph with a line break, writes the paragraph count to the appropriate flow
- For a paragraph with a line break, writes the offset of the hard line break to the appropriate flow
- For a paragraph with a line break, writes the paragraph text (255 chars, max) to the appropriate flow

NOTE: This command stores information only for one line break per paragraph. For example, assume your TOC refers to a heading that has a hard line break in the source document. When Maker generates the TOC, it includes that hard line break in the TOC entry. If you enter a subsequent hard line break in that entry, TocBreaker will not store it. Conversely, if you enter a line break *before* the generated one, TocBreaker will not store the generated line break. (To date, I have never had a TOC entry with two line breaks in it. Handling that possibility adds complexity to the dll, and since I initially made this as my own personal tool, I decided to keep it simple.)

3.3 About Applying Page and Line Breaks

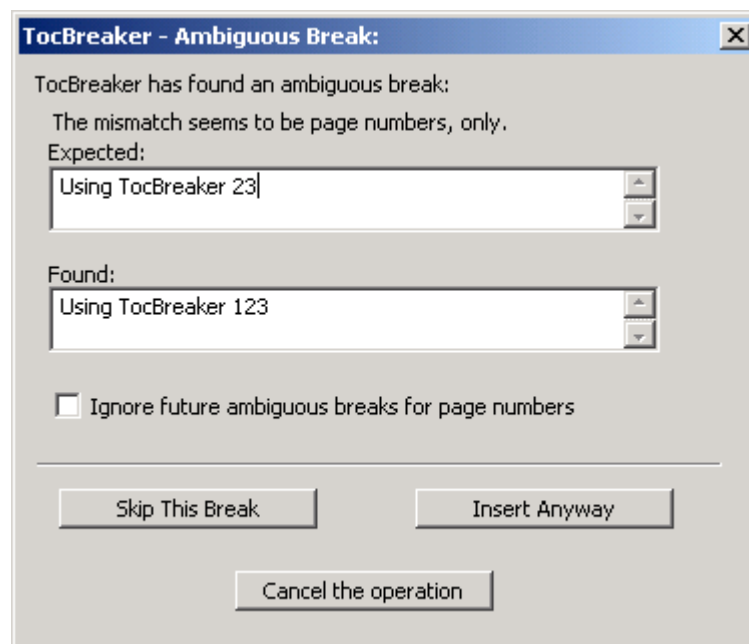
When you apply page and line breaks, TocBreaker loops through the stored information to identify which paragraphs should get the break, and what type of break to apply. It uses the stored paragraph number to identify which paragraph to process. It then uses the stored paragraph text to ensure that it has the correct paragraph. A mismatch in the text alerts TocBreaker that the order of paragraphs in your generated list has changed, and you may no longer want to apply a break to that paragraph.

There are two types of mismatches that TocBreaker can find:

- A pagenumber mismatch
A mismatch of the last word in the text. A word is any group of characters separated by spaces or hyphens (words are identified via the same algorithm used in the Word Count report). I assume the last word in a generated TOC or index entry is a page number. You can choose to accept all entries that have this type of mismatch.
- A main pgf text mismatch
A mismatch of any text that is not in the last word of the entries.

When you apply breaks to a file, TocBreaker displays a dialog box for every mismatch it finds. The dialog box is different depending on the type of mismatch, as follows:

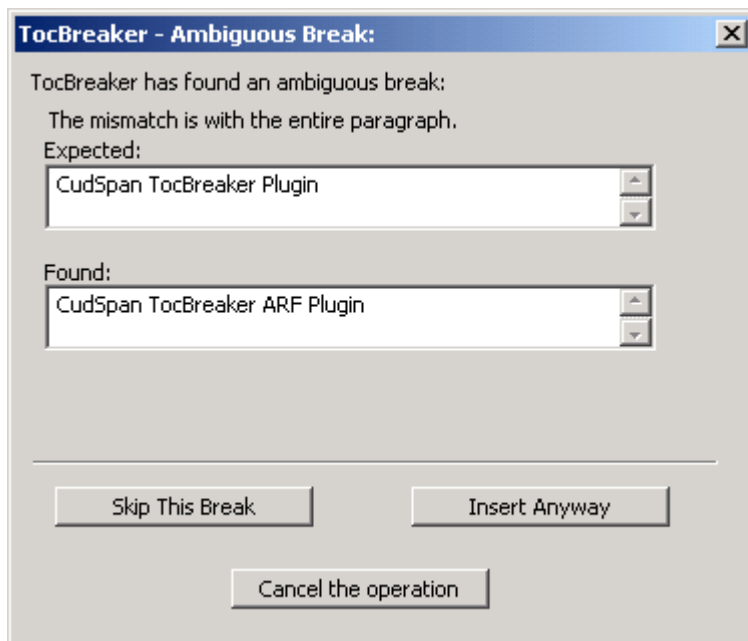
- A pagenumber mismatch



Notice that the expected entry is for page 23, while the current TOC entry is for page 123. You can skip the break and leave the TOC entry unchanged, insert the break, or cancel the operation.

You can also turn on the option to ignore future mismatches of this type. If you do, you can only choose to insert the break — the other options are dimmed. When you insert a break with this option turned on, whenever TocBreaker encounters this type of mismatch for the current generated file it automatically inserts the break anyway.

- A main pgf text mismatch



Notice that the expected entry is for *CudSpan TocBreaker Plugin*, while the current TOC entry is for *CudSpan TocBreaker ARF Plugin*. You can perform the basic operations — skip, insert, or cancel. But you cannot choose to ignore future mismatches of this type.

NOTE: Cancel simply dismisses the dialog box and stops the process. If you have already applied some breaks to the generated file, Cancel does not remove those breaks.

3.3.1 Applying Page Breaks

Choose Utilities > TOC Breaker Commands > Apply Page Breaks.

NOTE: There is no UNDO for this command. You should save your document before you use this command. If any problems arise, you can always revert to the last saved version of the file.

This command loops through each of the four page break info text flows. For each entry in each flow, it:

- Uses the paragraph count info to find the appropriate paragraph in the main flow
- Compares the first 255 characters of the main flow paragraph to the text written in the info entry
- If the text doesn't match, it displays a dialog box — you may either apply the break, go on to the next entry, or cancel the operation
- If the text matches, it applies the appropriate type of page break to the paragraph in the main flow

NOTE: This command only tests the first 255 characters in a paragraph. I assume that for generated files, this should be enough to ensure a unique paragraph.

3.3.2 Applying Hard Line Breaks

Choose Utilities > TOC Breaker Commands > Apply Line Breaks.

NOTE: There is no UNDO for this command. You should save your document before you use this command. If any problems arise, you can always revert to the last saved version of the file.

This command loops through the line break info text flow. For each entry in each flow, it:

- Uses the paragraph count info to find the appropriate paragraph in the main flow
- Uses the offset info to determine where to insert a hard line break character
- Checks the main flow paragraph for hard line breaks — if it already has one, and its offset is the same as the offset in the info entry, it ignores this entry and goes on to the next one
- Compares the first 255 characters of the main flow paragraph to the text written in the info entry
- If the text doesn't match, it displays a dialog box — you may either apply the break, go on to the next entry, or cancel the operation
- If the text matches, it inserts a hard line break at the specified offset into the paragraph in the main flow

NOTE: This command only tests the first 255 characters in a paragraph. I assume that for generated files, this should be enough to ensure a unique paragraph.
