JDF and XJDF Specification Editing

Date: 6th September 2016  
Author: Graham Mann  
Changes: Original created from original source documents

# Introduction

These notes were created by amalgamating and editing other instruction found in the 'readme' directory. They are written assuming the use of Dropbox to synchronize files between authors and CIP4's servers.

# Source Layout

## File Naming

Framemaker uses files of type 'book' as a main editing resource. These contain or reference multiple documents of type 'fm'. Each 'book' can reference the same documents and care should be taken (e.g. when updating) when making global changes not to affect other 'book' files.

In addition as part of the editing process Framemaker creates a number of housekeeping files for locking and backup. These are created in the main source directory and are thus normally copied to all uses via Dropbox. Files named like xxx.fm.lck are lock files for the source document xxx.fm; those named like xxx.backup.fm are working backups and those named xxx.recover.fm cropy up after a problem/crash etc. These and other files are used by Framemaker in a (usually fruitless) attempt to recover. These should be purged often.

Framemaker is capable of importing and flowing other types of document into its layout and this is used extensively for examples, diagrams and graphics.

## Directory structure

The source for the specifications is organized into the following arbitrary structure

* JDFspec-2015 Root directory
  + ArtOriginals Container for subdirectories
    - ?? Directory name roughly follows 'technology' used
  + builds Automated build results (nightly?) i.e. PDF's
  + FrameMaker Book and document source files ('\*book' and '\*.fm')
    - Examples Source for examples imported into '\*.fm' documents
    - Graphics Source for graphics imported into '\*.fm' documents
  + readme Notes and instructions

## Templates

These are special 'fm' files that are used to contain notes and more importantly formatting and setup values. They are used to globally set things such as character styles, spelling variations etc prior to updating a 'book' and producing a PDF. They can and should be edited albeit with care using Framemaker to effect changes to the various styles, done this way they will be propagated to all books.

It would be good if these were hierarchical - starting from a common root. As this is not the case changes must be manually made to all templates where necessary.

* TemplateForFinal.fm
* TemplateForFinalAllBlackJDF1x.fm
* TemplateForFinalAllBlackJDF2x.fm
* TemplateForFinalBOTH.fm

# Conventions

This section list the various editing conventions and styles.

## Spelling

The templates are capable of modifying the language of the underlying character styles.

When you run spell checking, maximize the checking to all but straight quote by depressing the “Options” button and then in the options dialog box:

* Check each box in the “find” options except “straight quotes”
* Leave all “ignore” options unchecked.

Depress the “Dictionaries” button and select the “Mark all paragraphs for rechecking” so that Framemaker checks all spelling. Otherwise, Framemaker silently ignores test that it shouldn’t.

Finally depress the “Start Checking” button The spell checking should find absolutely no errors unless some changes have introduced some errors. Spell checking should be run after a series changes to keep it as a useful tool that finds only a few problems each time it is run.

### Dictionary

The file **user.dct** (in ArtOrginals/user.dct) has all JDF terms and should be installed in the appropriate directory as the dictionary for a particular user, probably “Documents and Settings”/*username*/“Application Data”/Adobe/ FrameMaker/7.0/user.dct. The “learn” button in the spell checking dialog boxes adds words to this dictionary. The “Allow in Document” button adds the word to the dictionary within the actual Framemaker file.

Two caveats:

1. To avoid adding misleading plurals of JDF terms to the spelling dictionary, JDF terms should NOT use an “s” suffix to form a plural. For example, do NOT use QueueEntrys. Instead, use “QueueEntry elements”. Version 1.2 has an erroneous QueueEntries which was intended to mean “QueueEntry elements”. There are a few cases whether singular and plural JDF terms do exist, such as DevCap and DevCaps. The phrase “DevCap elements” is clearer than DevCaps because DevCaps is likely to be confused with DevCaps. The font change on the “s” is too subtle, and there are frequent cases in the specification where the “s” is in the same font as the JDF term by mistake.
2. Framemaker is unable to spell check words that exceeds about 25 characters (seems to be exactly 30 characters including hyphens for hyphenation points). Framemaker truncates the word in the dictionary and in the spell checker dialog box, Framemaker shows the suggested correct spelling as the truncated word followed by a blank and then the remaining letters of the word.

## Styles

The following sections describe the character, paragraph and table styles. In theory, no styles are overridden. In practice, some have been overridden, sometimes to make things fit and sometimes to avoid creating a new style.

The templates documented above have an example of all the styles used, refer to them for a definitive list. However, the following section show important styles and how they should be used.

### Character Styles

These styles are uses within a paragraph to highlight text of a particular type. Use with cross referencing to differentiate between references to resources, elements, messages etc.

**AnchorCalloutXXX:** used as a location for a box to hold bits of text surrounded by the XXX color

**Attribute:** a JDF attribute

**Bold:** makes text bold and disallows spell checking

**Bold\_italics:** makes text bold-italics and disallows spell checking

**Change\_Flag:** for blue text that specifies whether a feature is new, modified or deprecated feature

**ConformWord:** for all conformance words, such as MUST, SHOULD and MAY. The style forces the word to display in upper case

**crossRef:** not used.

**Element:** a JDF element

**Enumeration:** a JDF enumeration or value for an attribute

**EnumValuesAre:** used as a lead in to the paragraph detailing enumerations in tables

**Italics:** makes text italics and disallows spell checking

**Link\_External:** blue text for URLs

**Link\_XXX:** Many other types of character styles when cross referencing within the spec. The XXX is fairly self explanatory. If not add a note here

**Message:** a JDF message

**NoSpell:** sets language to “none” to prevent spelling checking from occurring, such as for “ ?” in the Name column and various other spelling anomalies

**NoteLeadIn:** start of a Note: highlights and bolds the character

**Process:** a JDF process

**Resource:** a JDF resource

**Sample\_Emphasis:** hardly used at all should be replaced with something else. Currently used to change JDF sample text to a color of “medium green”

**smallSize:** used to change text to 9pts in a few places to squeeze long words in to small spaces

**XPath:** used to format an XPath expression. This should be removed as it is mostly replaced by using combinations of Element and Attribute and Enumeration from above

### Paragraph Styles

**AnchorFigure:** used by Framemaker to make the insert point of figures. Deleting these can lead to format issues

**AnchorOnlySmall:** used for paragraphs that contain only anchors (for tables or figures). The paragraph is 2 pts so that it doesn’t take up much space. These paragraph are either in the main text and anchor a table or they are in a table and anchor a figure

**AnchorTable:** as for AnchorFigure but for tables

**AnchorTabularData:** as for AnchorFigure but for other stuff. These anchors appear to be used indiscriminately

**Body:** for second and subsequent paragraphs in the main text flow. Always preceded by a Body\_open\_paragraph paragraph

**Body\_open\_paragraph:** first paragraph after a section head or table

**Bullets:** bulleted paragraph that is flush left with body paragraphs

**BulletsN:** bulleted paragraph that is indented N levels from body paragraphs

**CalloutTitleReference:** title paragraph in boxed text; next to a Reference graphic

**CalloutTitleSummary:** title paragraph in boxed text; next to an Executive graphic

**CalloutTitleThought:** title paragraph in boxed text; next to a Thought graphic

**Callout\_paragraph:** paragraphs in boxed text

**Caption\_Figure:** style of each caption for a figure; always in a table of style Graphics

**Caption\_Table:** style of each caption for a table, always in a table of style Standard Table

**CellBody:** style of each cell of a Standard Table except for the header rows

**CellBody XXX:** same as CellBody, except that text has formatting XXX applied

**CellHeading:** style of each cell of a Standard Table header row

**CellHeadingRow:** style of a cell of a table that is important (bold); typically for cells in the leftmost column

**ChapTitle:** for the title of each chapter

**EnumDef:** for each paragraph that defines an enumeration; has a hanging indent

**EnumHeader:** introduces enumeration lists 'Values are:' type of thing

**FM\_Header:** for the title of each front matter chapter, such as the Preface. Also used heavily in the References appendix

**FM\_Ref:** used in describing references to other documents, such as in the References Appendix

**Footnote:** for footnotes

**HeadingN:** Nth section heading 2<=N<=7

**HeadingNOfElement:** Nth element section heading 2<=N<=7. Used for 'private' elements declared locally inside resources, messages etc. Automatically starts with word “Element”; allows for cross reference for element name in a table

**HeadingOfAttribute:** header for table specifically for the declaration of attribute values. Automatically starts with word “Attribute”; allows for cross reference for attribute name in a table

**HeadingOfExample:** introduces an example

**Heading\_Sub-head:** header without numbering; used in various places. Sometimes it seems arbitrary whether this style is used or a numbered subheading is used

**Numbered:** numbered list

**Numbered1:** first item of a numbered list

**Resource\_Body:** for Resource Properties (of a resource definition); second part and not bold

**Run-in-Head\_ResourceRef:**

for Resource Properties (of a resource definition), initial part. Each line in the Resource Properties consists of two parts: a paragraph whose style is Run-in-Head\_ResourceRef and a second paragraph (which appears on the same line, but tabbed to the right) whose type is eitherResource\_Body or Resource\_body. Even though the latter two styles are identical except for the latter being bold, they seem to be used in an arbitrary fashion with much overriding in use. The two styles should be merged into one and the overrides fixed

**Run\_in\_Header:** Not used - should be deleted  
used as run-in header in a various places; its use seems about as random asHeading\_Sub-head**.** Probably should have a more consistent use of headers

**Sample\_code:** for JDF examples; spell checking is turned off

**Sample\_codeNN:** for JDF examples; spell checking is turned off. Indented NN times

**TableFootnote:** some tables and figures have a footnote (usually spanning all columns)

### Table and Figure Styles

**Figure (was Graphics?):**A style for figures which defaults to one column and two rows, the first for the drawing and the second for a line of paragraph style Caption\_Figure. The rules are turned off. The use of the table to enclose the drawing and captions ensures that the two remain on the same page

**Standard Table:** A style for defining JDF elements, which defaults to one header row and a few body rows. The number of columns is usually either 2 or 3. The rules are dark blue and the header row has a dark blue shading. The table also has a title above the table with a style of Caption\_Table. Each such caption must include a “Section M of N” variable which conditionally displays when the table occupies more than one page

**Tabular Data:** A style for showing tabular data in a few parts of the spec. The rules are turned off

**Untitled Table:** Was used in the appendix for deprecated elements - now unused and not an option in Framemaker

#### Table Column Widths

The table column widths vary considerably, but always sum to 6.5”.

The strategy for column widths is described by the following rules:

1. The name column must be wide enough for all names to fit without a line break. Otherwise, (if they break) they are usually not searchable by Acrobat
2. The data type column must be wide enough for all names to fit without a line break. Otherwise, (if they break) they are usually not searchable by Acrobat
3. The Description column should be as wide as possible in order to keep the table as short as possible

For many tables, these three rules can be honored by using the widths of 1.4”, .85” and 4.25” (or something close to this) for the Name, Data Type and Description columns, respectively. For some tables, if rules 1 and 2 are honored, the Description column becomes very narrow and the table very long. The specification uses two work-arounds.

1. The data type column is made wide enough for most data types (.9” to accommodate NMTOKENS, but not for longer data types, such as IntegerRange­List, which needs 1.1”
   1. When names are extremely long, the name extends into the data type column (as a straddle), and the data type appears below in a separate cell. This results in a ruling override in the name column so that the name cell is shaped like an upside-down “L”. The tables with this work-around are:  
      LayoutPreparationParams, Media, PageList, PageData, StrippingParams and Document Properties

#### Table Row Heights

Sometimes table rows are quite long. Because Framemaker doesn’t break rows across pages, a large row can cause a large amount of white space on the page preceding the large row. Large rows are mostly caused by Attributes with a large number of values. When a large row would cause a large amount of white space, the Attribute values have been moved to a separate table with one value per row. The table row contains a reference to the attribute-value table and attribute is a hot link to the table as well.

# Importing Content

This include graphics text and examples. This is about to be quite heavily revised, but the technique remains much the same. Content is created in anther package, word, html, pdf whatever and Framemaker can import and display it. In the case of text it will attempt to 'flow' the text using whatever paragraph style is used, but (and it's a biggy) line endings have to be precise to avoid changes in paragraph styles - it's probably best not to use normal line endings and let Framemaker flow the text. If this is not possible ...

## Examples

TBD This needs to be re-written. The idea is to create all the examples as 'real' xml files and then to use a workflow to create rtf files with particular character fonts, colors etc.

## Graphics

Here are the original instructions for importing graphics. This too may be about to change.

...

See the file ArtOriginals/ListOfFigures.doc for a list of all figures in the spec and the corresponding original art file.

Here are the rules for creating graphics in the JDF spec

asdfa

1. Create the graphic in some format, such as Visio. MS Word or Adobe Illustrator This width should be a bit less that 6.5” or 165mm, i.e. about 6.4” or 162mm. This sizing avoid scaling in the spec and the resulting reduction in size of text
2. Save the graphic in the appropriate subfolder of "Art originals". The appropriate subfolders of Visio. MS Word or Adobe Illustrator are "Visio", "Word\_Art\_Conversion\_Source" and "AI", respectively. The "unknown" folder contains 2005 drawings in Visio that come from some unknown format
3. Create a png or jpg file from the original – png is best for anything that is not a photo. The file has the following characteristics
   1. It has the same name as the original except for the “png” or “jpg” suffix
   2. It is saved in the folder “Framemake 1.x/Graphics”, a parallel structure to the one in “Art originals”.When saving the png files, the following settings must be used:
      1. **Resolution:** set to "custom" with value of 150x150 dpi
      2. **Size:** set to “source”
   3. When saving the jpeg files, the following settings must be used:
      1. **Quality:** set to 90%
      2. **Resolution:** set to "custom" with value of 150x150 dpi
      3. **Size:** set to "source" if size would be about 6.5” or less or scaling in Framemaker is acceptable. Otherwise, click to “screen” and back so “source” to see actual dimension in “custom”. Then set the custom width to the desired size, e.g. a bit less that “6.5” and set the height to preserve the aspect ratio
4. In Framemaker, use the “files/import/file…” and select “import by reference”. Select a dpi of 150 dpi. This ensures that Framemaker doesn’t scale the 150-dpi-created jpeg. If necessary some adjustment may be require for fitting and/or quality

Note: some existing drawings have a larger jpeg drawing and allow Framemaker to do the scaling. The png drawings are all at the same scale as used by Framemaker.

# Build Instructions

Or how to create the PDFs. All versions use a common subset of Framemakers document files, depending upon which chapters are to be included in the final output. The build instructions require setting some conditional text view options for the 'book' as a whole - this will change those conditionals in the underlying 'fm' documents, and thus effectively for other books that include those documents.

For automatic builds this is not a problem as the source is copied to a private work area first. However, it a manual build is required using a public directory (i.e. Dropbox) then the user SHOULD restore the normal settings for editing. Essentially follow the instructions for creating the 'BOTH' version of the PDF last.

## PDF

These are general instructions to create a PDF. Any specifics for each version is noted in the succeeding sections. This is the complete list for the automatic build, some (many) of the steps may not be required for your build. If building either JDF or XJDF then repeat these instruction for Both and do this last. This will reset the 'Show/HIde Conditional Text' view for editing.

1. Update the Cover  
   This is required to change the title, date and version on the front cover.  
   These are edited in a word document, which is used to create a pdf file which in turn Framemaker includes in Cover.fm.
   1. Update the source document in ArtOriginal/MSWord  
      **Cover\_Art\_1.6.doc** and **Cover\_Art\_2.0.doc**
   2. Save as a PDF in FrameMaker/Graphics folder  
      **Cover\_Art\_1.6.pdf** and **Cover\_Art\_2.0.pdf**
2. Apply the correct template  
   This is required to ensure the correct conditional views are set. This is required prior to the updating of the TOC and LOF otherwise these will have the wrong content/format. It is also required prior to checking for broken cross references (if the 2.x spec references an item that is in 1.x only then for normal editing the references will not show as broken, however they will be broke in the final PDF as the content won't exist).
   1. JDF
   2. XJDF
   3. Both
3. Update 'book' contents  
   This is required to refresh the TOC and LOF etc correctly. It is also required prior to checking for broken cross references.
   1. Apply the correct template
   2. Update book
4. Fix any broken references  
   A necessary step prior to producing any release version.
5. Create the PDF  
   The object of the exercise after all.

### JDF

### XJDF

### Both

## HTML

This section has not be verified for 2015

### With Save As

1. Open the 'book' file with Framemaker
2. Open each document and do a “Save As”. Select appropriate html folder and then select “htm” file and change suffix to “html”  
   **Note**: Framemaker 9 has a bug whereby a Wingding conversion of “l” and “n” code points to special characters in Framemaker becomes pervasive for all fonts and thus turns all “l” and “n” code points to the special symbol even though the letter is not intended for normal fonts. For example the following are “and” and “all”: “a&#9632d” “a&#9679&#9679”.  
   **Note**: Framemaker supports “Save As” for a book. However, the JDF book is so big that Framemaker crashes before completing the conversion.
3. Using previous version as source and new html folder as target do the following
   1. copy index.html (main file)
   2. copy JDF.css
      1. it adds lines for putting colored borders around callouts
      2. It changes left margins for number (2), bullets (3) and ResourceBody
4. Using Framemaker folders:
   1. copy Cover\_Art.png from ArtOriginals\MSWord. The png file is produced by opening Cover\_Art.pdf with Adobe Acrobat and saving it as a png file
5. Run script fixHTMLDocs.bash (copied from previous version of html files)
   1. The script changes the style line to reference JDF.css
   2. check for correct conversion with compareHTMLDocs.bash
6. Edit Cover.html as follows:
   1. The “Save as” operation doesn’t include the pdf cover art and the Cover-1.png is too narrow.
      1. Change: <img src="Cover-1.png">  
         To  
         <img src="Cover\_Art.png" style="border: none;" alt="Cover\_Art.png" width="602"   
          height="785" border="0" />  
         <p><img src="Cover-1.png" style="border: none;"  
          alt="Cover-1.png" width="602" border="0"/></p>
7. Main file is index.html

### With RoboHelp

1. Select JDF1.book and start RoboHelp
2. Create new project. Current is JDF1.4a
3. Import Framemaker JDF1.book into new folder (htmlA2 currently)
4. Import files: JDF1LOF.fm, JDF1LOT.fm, JDF1LOE.fm and CoverBack.fm.
   1. Note: probably don’t have to do a Generate before this step, but didn’t try
5. In upper left pane, drag files under “HTML Files” into JDF1 folder
6. Edit file JDF1.4a/JDF1.4a.hhc by adding XML item Elements for Cover at beginning and for JDF1LOF, JDF1LOT, JDF1LOE and CoverBack at the end, using another item as a template or a previous version. These last two steps add the files to the HTML TofC
   1. Note: there is a similarly named file JDF1.4a\!SSL!\WebHelp\_Pro/JDF1.4a.hhc that the menu item Generate builds, apparently from the JDF1.4a/JDF1.4a.hhc file. There doesn’t appear to be a way to build this file via RoboHelp
7. Select properties of Layouts (lower left pane)
   1. Output should be within directory selected for import
   2. In Condition Build Expression field, exclude CaptionWContinued and CaptionWSheet so that table caption don’t have the continuation information for the case where the table is split across multiple pages
   3. Select Table of Contents as name of project, e.g. JDF1.4a
   4. Make default topic be “Cover.htm”
8. In RoboHelp pane, edit Cover.htm by copying previous versions cover, but add Cover\_Art.png which is missing because RoboHelp ignore referenced PDF files
9. Select “WinHelp”
   1. On right menu: Set as Primary Layout
   2. On right menu: Generate
10. Copy JDF.css from previous version into WebHelp\_Pro/JDF1
11. Run ‘fixHTMLDocs.bash’ which is a script that
    1. creates an “old” directory for source files of command run in this script file
    2. changse style references with ‘fixHTMLStyles.bash \*/\*.htm’
    3. checks if just style changed to “../JDF.css” with ‘compareHTMLDocs.bash \*/\*.htm’
    4. fixes overline errors in change flags with ‘fixHTMLDecoration.bash \*/\*.htm’
    5. fixes offset of a callout that has a large negative margin with fixIntroduction.bash
    6. fixes WingDings in the Hole appendix with fixHoleSymbols.bash
12. Remove “old” folder which has files before running above fixHTMLDocs.bash script
13. Move 9 files: Resources/Resources00203.jpg to Resources/ Resources00211.jpg from previous version to new version. The file generated by RoboHelp are wrong. These files came from the first version of html and then doubled in size to 1.6 x 1.1
14. Main file is “<project name>.htm”, e.g., JDF1.4a.htm: