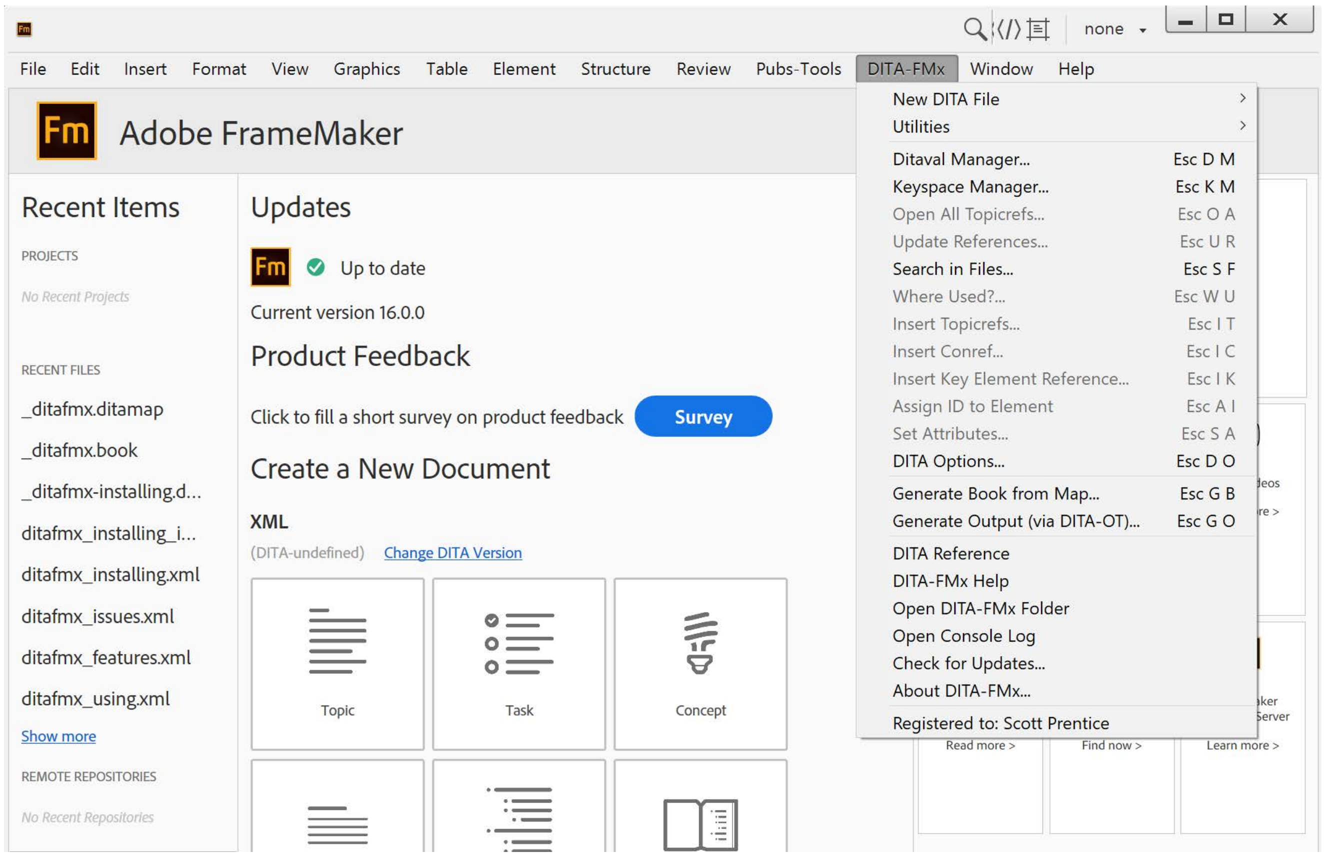


DITA-FMx User Guide

v.2.0.08
2020-10-11

Leximation, Inc.

DITA-FMx User Guide.
by Scott Prentice, Leximation, Inc.

Published by Leximation, Inc., 122 H Street, San Rafael, CA 94901.

This document was authored and published using FrameMaker and DITA-FMx.

The most current version of this guide is available on the Internet at
http://docs.leximation.com/dita-fmx/2.0/

Contents

1 Using DITA-FMx

Provides efficient and reliable authoring and publishing of DITA files with FrameMaker.

Documentation last updated: October 11, 2020

Updated for plugin client versions:

- authoring support client (*ditafmx_<fmver>.dll*) v.2.0.08
- import/export client (*ditafmx_<fmver>_app.dll*) v.2.0.08

RELATED INFORMATION:

"Features" on page 1
"Installation and Setup" on page 1
"DITA-FMx Commands" on page 1

Features

Describes the DITA map and topic authoring commands as well as the enhanced DITA publishing features.

DITA-FMx is a plugin and set of structure applications that let you create and edit DITA XML files in FrameMaker. DITA-FMx 2.0 supports DITA 1.2 (and earlier versions) as well as Lightweight DITA (as of v.2.0.06) and is available for FrameMaker versions 7.2 through 2020.

For a complete list of changes between versions of DITA-FMx, see Revision History. The following describes the general features provided by DITA-FMx.

DITA Map and Bookmap Support

A Map structure application is provided that allows for creation and editing of both DITA 1.2 map and bookmap files. When saved to disk, the resulting DITA map file is completely DITA-compliant, although within the FrameMaker authoring environment some additional elements are added to provide a more efficient authoring experience. This Map application also provides complete support for relationship tables as well as key definitions (keydef).

The **Build Map from Outline** command creates a DITA map and optionally DITA topic stub files from a simple FrameMaker file.

The **Build WorkBook from Map** command creates a FrameMaker book file that contains all of the files referenced in the current DITA map and all submaps. This “work book” is not intended to be used for publishing, but facilitates the use of FrameMaker’s built-in book processing commands such as spell checking and searching at the book level. In order to use these book processing commands on the work book you must first open all of the files in the book. This can be done with the **Open All XML Files in Book** command which provides the option to resolve references in each file, or open the files without resolving references.

Lightweight DITA Support

As of DITA-FMx 2.0.06, a set of LwDITA-FMx structure applications are provided, which are based on the current (as of July 2017) DTDs being developed by the DITA TC at OASIS. With these applications you can author and publish using Lightweight DITA.

Automatic Metadata Support

If enabled in the Auto-Prolog Options dialog, author and date metadata is automatically updated and added to the prolog in topics and bookmeta/topicmeta in maps.

When a <draft-comment> element is inserted, the author and time attributes are automatically set to the values defined in the Authoring Options dialog.

Ditaval Support

Filtering based on ditaval files can be applied to content in FrameMaker books and files using the **Apply Ditaval** command. This filtering can be done via conditionalization or by element deletion. Ditaval files can also be specified for output generated through the Open Toolkit (using the **Generate Output** command). The **Ditaval Manager** provides an easy to use interface for creating and managing ditaval files. A ditaval file can also be specified when generating a FrameMaker book using the **Generate Book from Map** command.

Key Reference and Keyspace Support

Keys provide an alternate method of referencing. A key name is bound to a reference using a key definition (<keydef>) in a map. This key name can be used in place of (or in addition to) an @href reference (a direct reference to a file). A keyspace is comprised of all key definitions in a map (and submaps), optionally filtered by an associated ditaval file.

DITA-FMx provides support for key references to files as well as key references to elements (key element references).

Glossary Term Keyref Support

Using a <term> element to create a key element reference to a glossary entry will render the text of the glossary entry's title (<glossterm> element). If you enable the **Glossary Term Swapping** option (in the **Book Build Settings** dialog), the term will render the <glossSurfaceForm> content in the first instance of the term in a chapter, and the value of <glossAlt> (<glossAcronym> or <glossAbbreviation>) for remaining instances.

Conref Support

Content references allow reuse of elements from the same file or other files on the same file system. DITA-FMx supports the use of conref ranges as well as conref specification by key reference (conkeyref).

If enabled (through the **Options** command) on the opening of a file, the content included by conref is resolved and displayed as a locked text range (similar to a text inset). The **Flatten Conref** command is available to "unlock" conrefs when saved as FM binary files.

Xref and Link Support

On the opening of a file, all xref and link elements are resolved and displayed as a locked text range. The auto-loading functionality may be enabled/disabled with the **Options** command.

When an <xref> or <link> element is inserted (from the element catalog), the Reference Manager dialog displays allowing you to select the target element for that element (through a direct reference or a key reference). Unless you enter text in the **Alternate Xref Text** field, the xref or link text will match that of the target element. The **External Xref** button lets you create an xref or link to an external file.

The **Xref to Hyperlink** command converts DITA-based xrefs and links into live hyperlinks in generated FM files.

DITA-FMx handles both DITA-based and FM-based cross-refs as both <xref> and <link> elements, for more information see Setting Up to Use Cross-References.

Coderef Support

The <coderef> element allows you to include the content of a non-DITA "code" file within a <codeblock> element. The **Coderef Manager** lets you create that reference as well as edit the content of the referenced file.

Attribute Management

The **Set Attributes** command provides quick and easy access to setting attributes on elements. This command makes use of the FrameMaker "Strings" attribute type and allows you to select one or more default values that are applied to the attribute. This is particularly useful with the DITA filtering attributes. This command also makes use of attributes defined as the "String" type and allows you to predefine a list of available values that are displayed as a scrolling list.

Support for Automated Publishing via FMx-Auto

DITA-FMx supports the FMx-Auto "addon" from Leximation to unlock the extended API features in DITA-FMx. This API can be used for automated publishing as well as other types of scripted or automated processing.

FM Book-based Output Support

The **Generate Book from Map** command builds a FrameMaker book from a DITA map or bookmap. It creates aggregated FM files from the top-level topic references and their child topicrefs (appropriate FM files are created from "part" files as well as frontmatter and backmatter files). Also offers the ability to include FM binary files in the generated book so you can mix DITA and unstructured FM files as needed. This allows you to generate a PDF book of an entire map. You can also pass this book to RoboHelp or Webworks ePublisher as one option for creating online Help.

Numerous options are available to automate portions of the book build, including the following:

- Add related-links from reltables as <link> or <fm-link> elements
- Convert <xrefs> and <links> into Hyperlinks
- Move <indexterm> elements in prolog to the topic title
- Apply ditaval "as conditions" or "by deletion" to perform ditaval-based filtering
- Move figure <title> elements to the end of a <fig> element
- Enable table <title> elements so they are used for tables that span multiple pages
- Flatten conrefs
- Assign numbering to the book component files based on their map element type
- In a bookmap, replace the "list" files with generated FM lists

- Apply templates to the book component files based on their map element type
- Run one or more custom FDK client, ExtendScript, or FrameScript automations
- Update the generated book and files

DITA Open Toolkit Integration

DITA-FMx integrates with all versions of the DITA Open Toolkit. Versions prior to 2.0 require additional setup, but recent versions are virtually “plug and play.” Version 3.2 or later allows use of the Dynamic-Targets option which automatically lists all available publishing targets.

The **Generate Output** command provides the ability to run a specific target in an Ant script to generate output through the DITA Open Toolkit. One option lets you use a provided Ant script to generate output based on the current file (a topic or map), or another option lets you select a target in an Ant script that you provide. Using the Current File option, you can specify a ditaval file for filtering. For more information see, Generate Output.

DITA2Go Integration

DITA-FMx includes integration with DITA2Go (from Omni Systems at dita2go.omsys.com) as an alternative to the DITA-OT for creation of various types of online Help. If DITA2Go is installed, the “DITA2Go Project Manager” menu item will be available on the DITA-FMx menu.

Locating Content in Files

The **Search in Files** command lets you search for content in files within a folder (and sub-folders) or in files referenced by a DITA map. The search criteria can be a mix of textual content, element or attribute names, or attribute value.

The **Where Used** command generates a report listing all files that reference the selected element or current topic.

File Management

The **Reference Report** command generates a report of all referenced files in a DITA map or topic file (as well as all references in all referenced files). This report can list unresolved and/or resolved references, as well as specific reference types (topicref, xref/link, conref, and image).

The **Create Archive** command generates a ZIP archive of the current file and all referenced files.

Support for FrameMaker Variables

If enabled in the **Authoring Options** dialog, FrameMaker variables are wrapped in <fm-var> elements, which round-trip to DITA as <ph>

elements. The variable name is preserved as a special value in the outputclass attribute which allows it to round-trip successfully.

The "book-build" process provides the ability to import metadata (attributes and content values) from the map into the generated FM files as variables. This lets you update variables in the headers and footers as well as variables in binary FM files (like those used for the title page and other frontmatter).

Support for Graphic Overlay Objects in Anchored Frames

Textual callouts and graphic objects within an anchored frame will now round-trip from FrameMaker to DITA and back. The data to define these objects is stored in the DITA <data> element.

Options

An **Options** command provides the ability to specify the structure applications for DITA map and topic file authoring, the structure application used for the book processing, as well as control of various DITA-FMx options.

Context-Sensitive Help on DITA Elements

You can get context-sensitive help for DITA authoring by pressing Alt+F1. The DITA Reference displays the topic that relates to the element currently selected. If you have added elements through specialization, you can add information about your elements to the CHM file (the source is provided in the DITA Open Toolkit).

Specialization

DITA-FMx should fully support specialization (or at least not hinder it). If you have a specialized data model, you will need to make the parallel changes to your DITA EDD and r/w rules. The only effect of specialized elements is with regard to element names, and the only place DITA-FMx operates solely on element names is with the processing of tables (and in this case, additional table elements can be defined in the *ditafmx.ini* file). In all other cases, DITA-FMx processes elements based on their class name, so it should properly handle specialized elements.

Table Support

DITA-FMx fully supports all DITA table types including simpletable-based tables. Users who create specialized tables can ensure that they are properly rendered by including them in the Element Mapping dialog via the Options dialog.

DITA-FMx provides support for complex table formatting through the use of predefined table formats as well as preserving table widths and the state of rotated table cells, in addition to indenting tables based on the left indent of the parent element. Custom ruling and shading can be applied to rows and cells by preconfiguring a reference page table and assigning outputclass attribute values.

Hazardstatement Support

DITA-FMx provides special support for authoring and publishing of <hazardstatement> elements. When using the default Topic structure application, the<hazardsymbol> element will move into a sensible location while authoring. When publishing with the default Book structure application, the <hazardstatement> elements will be wrapped in a table structure with a special header and coloring based on the hazard type (attribute). For additional information on this feature, see DITA-FMx Specific Book Application Elements.

Indexterm Support

On file open, DITA-FMx converts <indexterm> elements to a FrameMaker-compatible syntax within Index markers. DITA specifies that index subentries are defined by nested <indexterm> elements. This feature collapses nested <indexterm> elements into a single semi-colon-delimited string within the top-level <indexterm> element which can be properly interpreted by FrameMaker and converted into an Index marker. This functionality keys off of the value of the class attribute, allowing it to work for specialized instances of the <indexterm> element. On file save, the Index markers are converted back to valid nested <indexterm> elements. If necessary, in order to support more complex element structures within <indexterm> elements, you can disable the conversion to markers.

Reference Support

All textual references (topic references, conrefs, xrefs, and links) are represented in FrameMaker as locked text ranges similar to text insets. These text ranges are not linked to text flows but are used as a means to lock a region of text and allow the user to click on the object. In order to maintain valid DITA files, DITA-FMx converts these text ranges to the appropriate XML structure on file save.

Localization Support

DITA-FMx provides a mechanism to apply language-specific templates based on the topic/@xml:lang attribute value. This attribute value can also be used to specify language-specific book-build options. For more information see the “UseLanguageTemplate” parameter in INI-Only Settings.

CMS Support

DITA-FMx seamlessly integrates with the Bluestream XDocs CMS through the XDocsFMx plugin (available at no cost). Customers have also had success with integrations of DITA-FMx and SDL Trisoft as well as Vasont. We also have development prototypes to integrate DITA-FMx with the easyDITA CMS.

Using FMx-Auto, DITA-FMx can be set up on the back end of most CMSes and used as a PDF publishing engine. To learn more about this support and options visit www.leximation.com/dita-fmx/cms-support.

RELATED INFORMATION:

"DITA-FMx Commands" on page 1
"Revision History" on page 1
"Limitations" on page 8

Limitations

Known limitations in the current version of DITA-FMx.

The following list describes the currently known issues that apply to using DITA-FMx on all supported versions of FrameMaker (see below for version-specific issues):

- DITA 1.3 is not currently supported by DITA-FMx. We are assessing which features to support; if you have specific needs, please contact us.

  While no support is provided out of the box, there is no reason that DITA-FMx would not support the DITA 1.3 model if an appropriately defined structure application were developed. The aspects of DITA 1.3 that will not currently work, involve new methods for ditaval filtering, scoped key resolution, and cross-deliverable linking.

- The following DITA 1.2 features are not currently supported:
  - keyrefs to topicref-based elements are not supported
  - key element references to <fm-indexterm>, <indexterm>, and <indextermref> are not supported
  - key element references will provide the defined content but will not currently provide a hyperlink to a provided @href target
  - inline elements within key element references are unwrapped; the content will remain, but the markup is deleted
  - text from conrefs and keyrefs within a map will not be passed to the book-build process
  - the @conaction attribute is not supported
  - a “Learning” structure application (based on the Learning and Training specialization) is provided for testing purposes only and is

not considered to be production-ready; an update will be provided in a future release

  – a “SubjectScheme” structure application is not currently provided

- When generating an FM book from a map, you may get the following messages in the FrameMaker Log file:

  – XML Parser Message: **“ID '*TOPICID*' has already been used”**

    This message is caused when your topic files contain duplicate IDs. This can happen when you clone a topic file to create a new topic file or if you include the same topic in multiple places in the map. Both of these situations are valid as far as DITA is concerned, but FrameMaker doesn’t like to see duplicate IDs in a single book. This is just a warning and can generally be ignored.

    The only situation where this duplicate ID issue will cause problems or possibly unexpected results, is when you have xrefs or related links to a topic where the ID exists in multiple places in a given chapter file. The reference will always end up pointing at the first instance of the ID in a generated FM chapter file.

- For the **Apply Ditaval** command, only the <prop> ditaval element is supported. Other elements may be recognized in a future release if that is seen as beneficial.

  The *as conditions* option of the **Apply Ditaval** command does not necessarily result in filtering that matches that of the DITA Open Toolkit. The conditions are applied properly based on the filtering attribute values, but the default hide/show logic of FrameMaker conditions is not the same as that used by the OT. When using the *as conditions* option, it is best to use ditaval files that only use the “exclude” action attribute; this will provide more reliable results. Due to the nature of conditional tagging, this option cannot remove components from a book. If this is needed, use the *by deletion* option.

  The *by deletion* option of the **Apply Ditaval** command, more closely emulates the filtering done by the DITA-OT. However, the *by deletion* option only supports the “include” and “exclude” action attributes; the “flag” action is not supported (if “flag” is needed, use the *as conditions* option).

- Because a graphic object in FrameMaker cannot have children, the <alt> and <longdescref> elements are not fully supported. The text of the <alt> element will round-trip and can be edited through the Object Properties (Object Attributes) dialog, but any attributes on the <alt> element will be discarded on file open. The <longdescref> element will be discarded on file open.

- When using TextLine graphic overlay objects, do not use a Center or Right alignment, only use the default Left alignment. Using an alignment of Center or Right will result in the text line position shifting to the left or right when the file is reopened. This will be fixed in a future release.

- In a relationship table, only @href references to files are honored. Any @href attributes that include references to topic IDs will be stripped down to just the file name for processing and creation of related links. This may change in a future update.

- DITA-FMx does not support the “use by reference” feature of DITA footnotes. This is where the footnote reference number only shows up at the referenced (via <xref>) location but not at the location of the <fn> element. In all cases, there will be a reference number generated by the <fn> element. You may add references to that footnote with a formatted reference (<fm-xref>).

- Indexterms with multiple child elements that are siblings, will not round-trip properly. On import, the sibling <indexterm>s are incorrectly converted into FM index syntax, which means the exported <indexterm>s will be nested rather than siblings. A workaround is to disable <indexterm> conversion on import and export (although this will not allow you to generate a FM index). This may be fixed in a future update.

- Xrefs within the link/desc element will not resolve properly on file open. After file open, running the Update References command will resolve these references. This may be fixed in a future update.

- Backslashes used in an external <xref> (either in the element text or in the @href value) will not render as expected. You should use forward slashes if at all possible. This is due to a FrameMaker limitation and cannot be resolved.

- The **Generate Book From Map** command disallows the building of a book from a DITA map that references files on multiple disk drives. This appears to be a core FrameMaker limitation (but this probably isn’t a good idea anyway).

  When generating a book from a map, any attributes on topicref-based elements that reference a submap are lost during the XSLT import process. The “merge” process replaces the *referencing* topicref with the *referenced* topicref (the root topicref in the referenced map). Therefore, any filtering (or other) attributes intended to affect the generated book, should be placed on the *referenced* topicref rather than the *referencing* topicref.

- If an XML file is “pretty-printed” and a line breaks after an inline element (such as a <ph>), when opened in Frame the space between that element and the following word will be lost.

- Child elements within an <xref> or <link> are lost on import if the Auto-Load Xrefs option is enabled. If your xrefs contain child elements, disable this option.
- Conrefs within titles won't be included when you run the Update References command in a DITA map unless the target file is already open. If you have conrefs in your titles, you should open the file first before running the Update References command.
- Deleting an <fm-reflabel> element from a DITA map file without deleting the entire <topicref>, may result in the leading symbol being left in the map. This is a temporary issue and will go away the next time you open the file; it has no effect on the ability to process the files.
- The <syntaxdiagram> child elements should round-trip properly, but the formatting of these elements may not be optimal. We will focus on providing better support for these elements in DITA-FMx 2.x.

FM 2017 Issues that relate to using DITA-FMx with FrameMaker 2017:

- FM version 14.0.2 or later is required for use with DITA-FMx due to FDK library updates with that update.
- The FM 2017 "welcome screen" has no API, so the DITA-related links and buttons will not work when DITA-FMx is installed. When Adobe provides an API to these items, we'll hook it up.

FM XMLA Issues that relate to using DITA-FMx with FrameMaker XML Author (FM 12 and 2015):

- DITA-FMx does not work well with FrameMaker XML Author. Many commands work fine, but due to the inability to open FM binary files, you cannot create new DITA files through the DITA-FMx interface and the publishing features will also not work. You can work around this limitation by opening an existing XML file and saving it to a new name (remember to recreate the topic ID). You can also use DITA-FMx to author files that are stored in a CMS such as XDocs or easyDITA.

FM11 Issues that relate to using DITA-FMx with FrameMaker 11:

- Due to changes in the FrameMaker 11 API, the "Open Maps in Document View" option does not work on the initial open of a map. It will however, work when opening a supmap from a topicref in a map. If a solution is found it will be provided in a future update.
- Be aware that using the FrameMaker 11 Code View can strip leading spaces from preformatted elements (i.e., <codeblock>, <lines>, <msgblock>, <pre>). This is a core FM bug, that will hopefully be fixed soon. Unfortunately, there's nothing that DITA-FMx can do about this (other than to warn you about it).

FM10 Issues that relate to using DITA-FMx with FrameMaker 10:

- FrameMaker 10 includes a number of "Save Ditamap As" options that are not functional when DITA-FMx is installed (because the FM10 DITA support is uninstalled). The following SaveAs options are not supported:
  - Composite Document
  - Book with FM Components
  - PDF
- The two left-most buttons on the Resource Manager (map editor) toolbar are non-functional when DITA-FMx is installed. We are looking into options for enabling them, but for now you must use the element catalog to insert topic referencing elements (<topicref>, <chapter>, <appendix>, etc.). The remainder of the buttons seem to work properly with DITA-FMx.

FM9 Issues that relate to using DITA-FMx with FrameMaker 9:

- FrameMaker 9 includes a number of "Save Ditamap As" options that are not functional when DITA-FMx is installed (because the FM9 DITA support is uninstalled). The following SaveAs options are not supported:
  - Composite Document
  - Book with FM Components
  - PDF
- The four left-most buttons on the Resource Manager (map editor) toolbar are non-functional when DITA-FMx is installed. We are looking into options for enabling them, but for now you must use the element catalog to insert topic referencing elements (<topicref>, <chapter>, <appendix>, etc.).

FM8 Issues that relate to using DITA-FMx with FrameMaker 8:

- A bug exists in FrameMaker 8 (and FM7.2) that causes it to crash if more than approximately 350 XML files are opened in the same session. In normal authoring use this limitation does not cause any problems, but if you are publishing a book that contains more than 300 topics, it is very likely that you'll run into this problem. The only workaround is to convert your maps into a book in multiple segments and assemble the final book once all of the components have been built (restarting FrameMaker in between each build). DITA-FMx displays a warning after opening more than 300 files to remind you to restart FrameMaker. This bug has been fixed in FrameMaker 9.

FM7.2 Issues that relate to using DITA-FMx with FrameMaker 7.2:

- In the FM7.2 version of DITA-FMx, double-byte characters are garbled in the fm-topicreflabel elements, and don't update properly.

Using DITA-FMx for DITA authoring in binary FrameMaker files is not recommended. Many of the features of DITA-FMx rely on the files being XML (allowing them to be parsed on disk). The following are reported issues, but there may be others:

- A file that can't be opened due to missing fonts or images and it is the target of a conref or <xref>, the Reference Manager won't display when the conref or <xref> is double-clicked. This can be resolved by opening the referenced file before double-clicking the conref or xref.
- Double-clicking a conref then choosing Update, will delete the conref.

Please send any problems or suggestions to <ditafmx-help AT leximation DOT comditafmx-help@leximation.com>.

RELATED INFORMATION:

"DITA-FMx Commands" on page 1
"Features" on page 1

Tips and Troubleshooting

Tips for making the most efficient use of DITA-FMx.

The following list describes common and useful tips for working with DITA-FMx. For additional information, please visit the FrameMaker/DITA Community KB at kb.leximation.com/dfm/.

If you have tips or suggestions you'd like to share, please send them to <ditafmx-help AT leximation DOT comditafmx-help@leximation.com>.

Generate Book from Map command fails with XSLT Processor message.

The FM 13.0.1 update changed the default XSLT processor from XALAN to SAXON. If you're using the latest (as of v.2.0.06) default Book app, with FM12 or earlier, you'll need to manually switch the XSLT script to XALAN (see, Special Instructions for FM12 and Earlier). If you're using a custom Book app on FM 2015 or later, you may need to update your XSLT import script to work with SAXON (or change the default XSLT processor to use XALAN).

The error message: `Can not convert #RTREEFRAG to a NodeList!`

“Zapfdingbats Font Family not available” message in console.

If you’re seeing this message in the FrameMaker console window, you’ll need to modify the *entfmts* file that ships with FrameMaker. This file is found in the FrameMaker\Structure folder, and is a FM binary file (although it doesn’t have a “.fm” extension). To fix this annoyance, open that file in FrameMaker (make a backup first), and change the Font Family of the “FmDingbats” character style to “Adobe Pi Std”. While you’re in there you may also want to change the default font family of the “Bulleted” paragraph style from “Courier” to “Courier New” (if you’re seeing console messages about Courier). Save the file. (Make sure it remains named “entfmts” not “entfmts.fm”.)

If you’re seeing other font messages in the console, that may be due to other fonts used in that file or template files. The easiest way to locate the problem is to save the template(s) to MIF and search in the MIF file for the offending font name.

Spelling Checker - Allow in Document.

By default (in DITA-FMx), when you spell check a file and use the “Allow in Document” option, that word is stored as a processing instruction in the DITA file. For words that are common to an entire project, you might consider using the “Learn” option instead, which stores the word in a system-level location. To disable this feature (prevent creation of the dictionary PIs), change the GeneralExport/WriteDictionaryPIs parameter to “0” in the *ditafmx.ini* file.

Supporting round-tripping of image sizes.

To support the proper sizing and placement of image elements, certain read/write rules must be defined. If you’re using a custom or older structure application, the height and width rules may not be properly defined. Refer to the image topic for the proper rules.

Images aren’t “shrink-wrapped” in book builds.

If your images resize and shrink-wrap properly while authoring, but don’t for your book builds, check the Reload References setting in the Book Build Settings dialog. This setting must be enabled for images to resize and shrink-wrap in book builds.

Making use of page-wide tables.

If you want a table to extend to the full width of the text frame, set the table/@pgwide attribute to 1.

Using FrameMaker variables with DITA

As of DITA-FMx 2.0 variables (user and system) will round trip properly if the “Convert Variables to fm-var on Save” option is selected in the Authoring Options dialog.

NOTE: Using FrameMaker variables is not really considered to be a "best practice" although there may be situations where it is an ideal solution for a particular problem. The proper DITA way to use "variables" is that of a conref to a phrase element or as a keyref with a "key element reference."

Heavy use of references slowing things down?

If you make heavy use of references (conrefs or xrefs), you may find it more efficient to open the target files first (those that are the destination of an xref or the source of a conref). If the target files are already open when you open topic files, the referencing process will go much faster.

Reference problems while converting unstructured content to DITA

While converting an existing set of unstructured files into DITA, you may want to disable the auto-loading of xrefs and conrefs. If auto-loading is enabled you may get a lot of referencing errors when opening files if the target of those references is not a completely valid file.

Conrefs in title elements

If you have titles that contain conrefs, be sure to have that file open when updating the DITA map file, otherwise the conref content will not appear in the topicref label.

Use of inline formatting within "preformatted" elements (like codeblock)

FrameMaker uses the read/write rule "preserve line breaks" to allow the line breaks within code or preformatted elements to round trip between XML and the authoring view. The use of inline child elements such as <b> or <i> within a preformatted element poses a particular problem since you generally don't want line breaks preserved for those child elements when used in non-code situations. There are a number of ways to handle this problem, but the easiest is to only apply these inline elements within a line (don't tag multiple lines), and don't let the child element start at the beginning of the line (allow at least a leading space before the inline element starts and ends). What appears to be a Frame bug causes the preserved line break to be lost if a child element starts or ends a line.

Use of draft-comments inline

If you make use of the <draft-comment> element within running text, be sure to wrap a trailing (or leading) space within the element so that when (or if) you conditionalize these elements so they are hidden in Frame, you don't end up with a double space.

Quick way to add a row to tables

Place the cursor anywhere in a row, press Ctrl+Enter and a new row is added after the current row.

Converting unstructured to structured files

When creating a conversion table, be sure to map Index markers to a valid element type. In the default DITA-FMx Topic app, the <indexterm> element is not defined as a Marker type, but a Container. Index markers should be mapped to the <fm-indexterm> element instead because that is defined as a Marker element type in the EDD. If you map Index markers to the <indexterm> element, FrameMaker will crash with an assertion failure error when saving the structured file to XML.

RELATED INFORMATION:

"image" on page 35
"Working with Images" on page 26
"DITA Options" on page 33
"INI-Only Settings" on page 51

Using Lightweight DITA in DITA-FMx

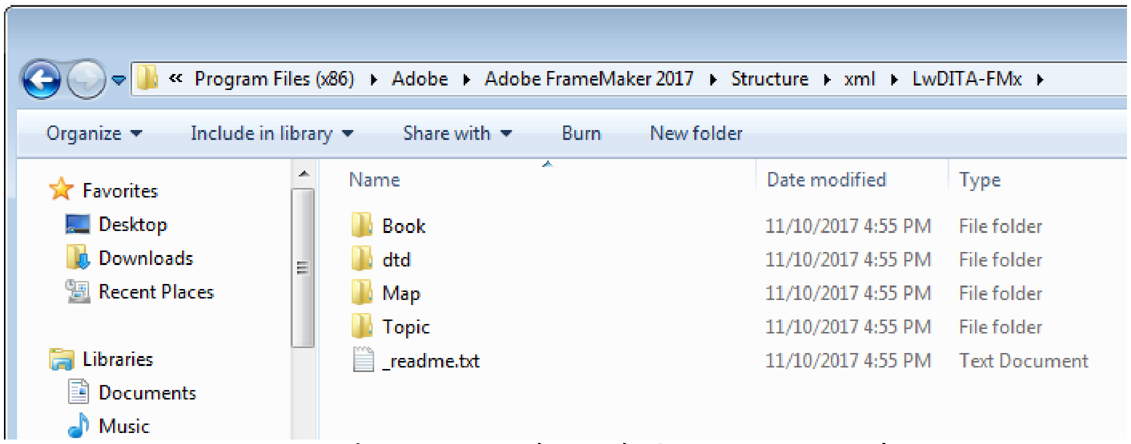
Lightweight DITA offers a simpler model for authoring of content.

As of v.2.0.06, DITA-FMx supports authoring and publishing of the Lightweight DITA model. While the specification for this model is not final, it seems stable enough to provide a set of (beta) structure applications to give users a taste of this alternative authoring and publishing option.

The structure applications provided with DITA-FMx are based on the model as of June 2, 2016. Updates to the DTDs are available on github.com.

REMEMBER: *These structure applications are considered to be "beta" samples. If you run into problems or have any questions, please contact us at <ditafmx-help AT leximation DOT comditafmx-help@leximation.com>.*

In order to use the Lightweight DITA model, you need to install the structure applications. We provide the same set of apps as with the full model, Topic, Map, and Book. These apps are available in *LwDITA-FMx_apps.zip* file found in the DITA-FMx installation folder (*FrameMaker\DITA-FMx-2*). Just extract the contents of that file into your *Structure\xml* folder. This should result in a LwDITA-FMx folder with the application folders inside.



Name	Date modified	Type
Book	11/10/2017 4:55 PM	File folder
dtd	11/10/2017 4:55 PM	File folder
Map	11/10/2017 4:55 PM	File folder
Topic	11/10/2017 4:55 PM	File folder
_readme.txt	11/10/2017 4:55 PM	Text Document

Figure 1-1: Lightweight DITA structure applications

Each of the application folders contains a “structapps-stub” file along with the required application files. Insert the stub file into your structure application definition file as described in Manual Installation of the Structure Applications (using the LwDITA stub files instead of the default app files).

After installing these applications, set the Book, Topic, and Map applications to the LwDITA apps in the DITA Options dialog. Once this is complete, you should be able to create and edit Lightweight DITA files in FrameMaker.

The *_readme.txt* file contains additional information about Lightweight DITA and the provided structure applications.

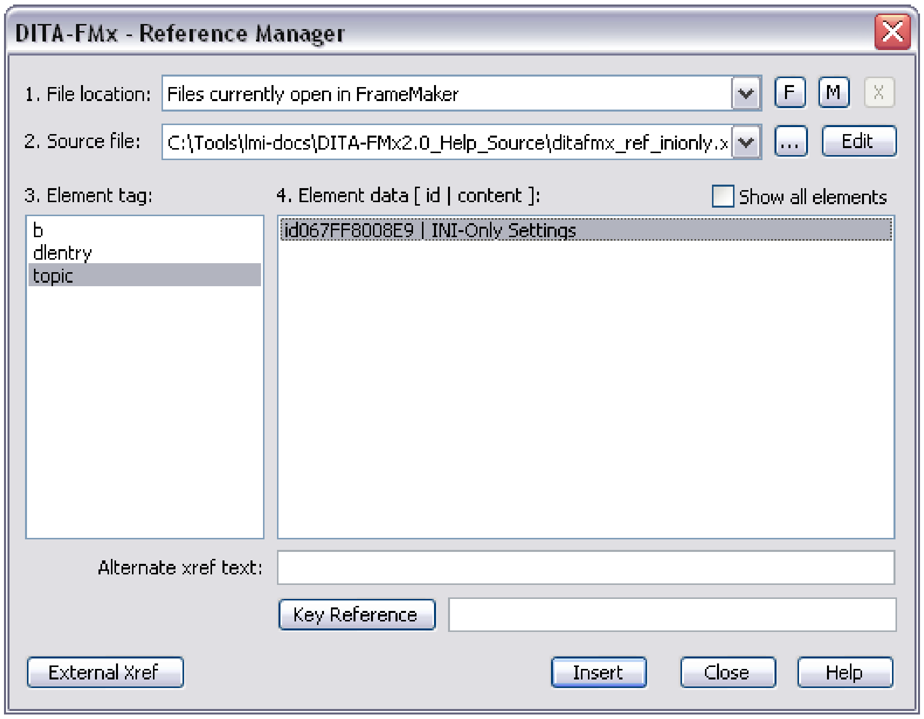
RELATED INFORMATION:

"Manual Installation of the Structure Applications" on page 13
"DITA Options" on page 33

Using the Reference Manager

Lets you select a conref, xref, or link target by specifying the file, element type, and element.

The Reference Manager is displayed when inserting a conref, <xref>, or <link>. To insert a conref, choose **Insert Conref** from the DITA menu, to insert an <xref> or <link>, use the Element Catalog. Note that the Reference Manager is only displayed for <xref> or <link> elements if they are defined as a “Container” rather than a “Cross-Reference” element in your EDD.



1. File location: Files currently open in FrameMaker
2. Source file: C:\Tools\lmi-docs\DITA-FMx2.0_Help_Source\ditafmx_ref_inionly.x
3. Element tag: b, dlentry, topic
4. Element data [id | content]: id067FF8008E9 | INI-Only Settings

Figure 1-2: DITA-FMx Reference Manager

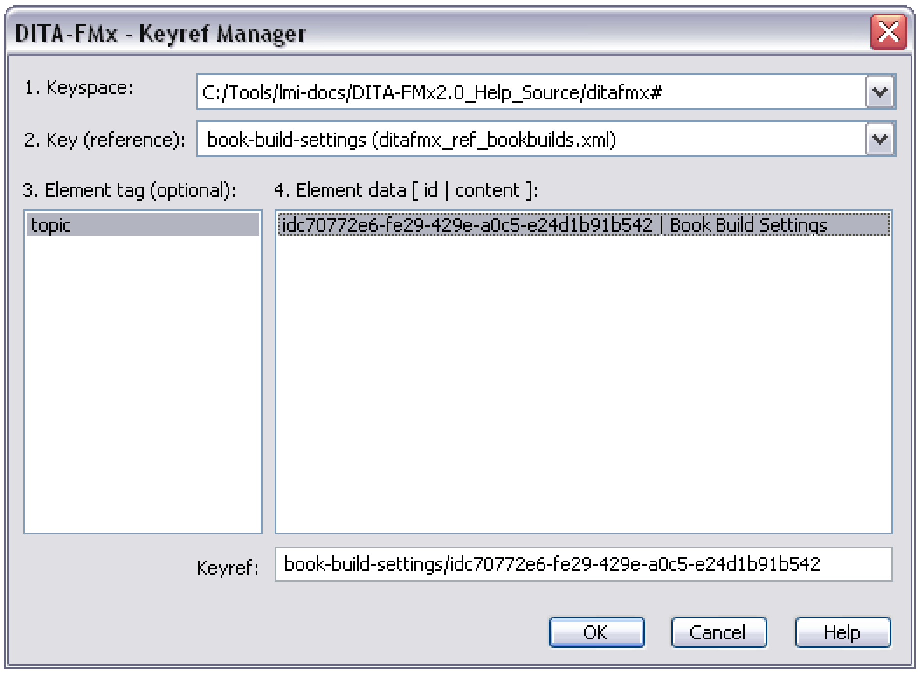
To insert a reference of any kind, first specify the file that contains the reference source. To select a source file, specify the file location using the File Location list. This list will initially only contain the label “Files Currently Open in FrameMaker” but you can browse through files on disk (without opening them first), by choosing the “F” button (select folder) or the “M” button (select map).

Selecting “Files Currently Open in FrameMaker” will display those files in the Source File list. Selecting the “F” or “M” buttons will list all “.xml” and “.dita” files in that folder or map. If you want to place a reference to a file that is not open, use the Browse button (“...”) to open another file.

By default the Element Tag list displays only elements that have id attribute values in the source file. For conrefs, this list is further restricted by displaying only elements that are valid at the current insertion point. If you want to limit the xref or link targets to a specific list of elements, use the XrefElements parameter in the *ditafmx.ini* file (for details, see INI-Only Settings). When you select an element tag name from the list, the available target elements display in the Element Data list. By default, only elements that have an ‘id’ value are available, but if you select the Show All Elements option, you will be able to select from all available elements of the selected type (this option is only available if the “Files Currently Open in FrameMaker” file location is selected).

The elements are listed with their ‘id’ and textual content. To place the reference, select an element and choose the Insert button. If you have selected an element that does not have an ‘id’ value, you will be prompted to provide an ‘id’ for that element (the ‘id’ is written to the source file, so be sure to save that file before exiting). If you want to specify text for the xref that is different than that of the target element, enter that text in the Alternate Xref Text field. To insert an xref to an external file, choose the External Xref button.

To insert a reference based on a key, choose the Key Reference button (or enter the key value in the Key Reference field). The Keyref Manager dialog allows you to select a key from the specified keyspace, then optionally select an element within the file referenced by that key. When placing an xref or link to the root topic in a file, it is only necessary to provide the key name. When inserting a conref, you should provide the key and the sub-element reference.


DITA-FMx - Keyref Manager

1. Keyspace: C:/Tools/lmi-docs/DITA-FMx2.0_Help_Source/ditafmx#

2. Key (reference): book-build-settings (ditafmx_ref_bookbuilds.xml)

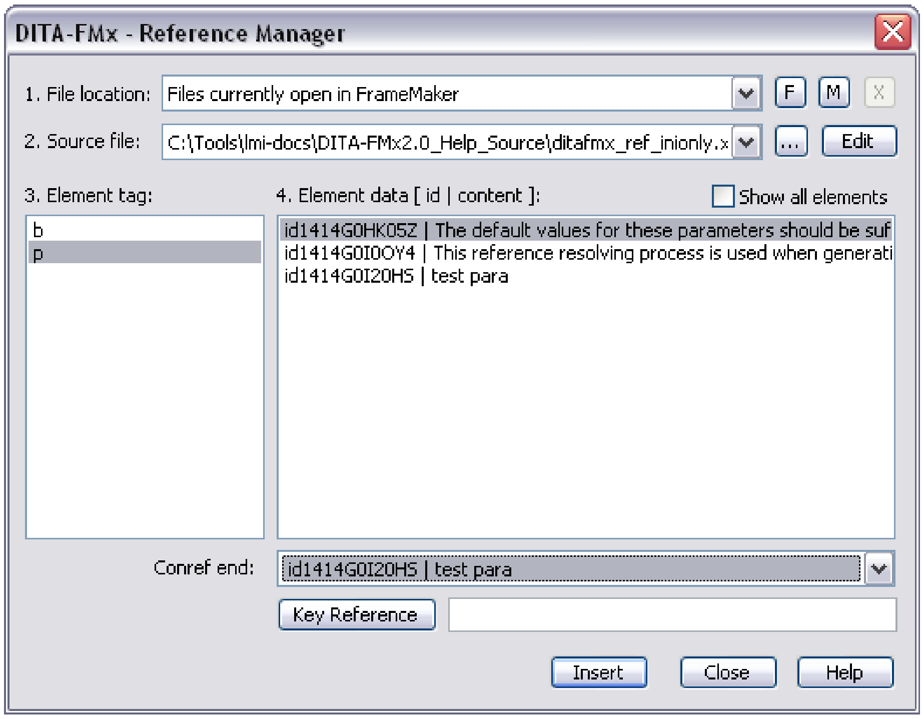
3. Element tag (optional): topic

4. Element data [id | content]: idc70772e6-fe29-429e-a0c5-e24d1b91b542 | Book Build Settings

Keyref: book-build-settings/idc70772e6-fe29-429e-a0c5-e24d1b91b542

Figure 1-3: DITA-FMx Keyref Manager dialog

When inserting a conref, the Alternate Xref Text field is replaced with a Conref End list. This is populated with valid conref end targets based on your selection in the Element Data list.


DITA-FMx - Reference Manager
1. File location:
Files currently open in FrameMaker
F
M
X
2. Source file:
C:\Tools\lmi-docs\DITA-FMx2.0_Help_Source\ditafmx_ref_inionly.x
...
Edit
3. Element tag:
4. Element data [id | content]:
Show all elements
b
p
Conref end:
Key Reference
Insert
Close
Help

id1414G0HK05Z | The default values for these parameters should be suf
id1414G0I0OY4 | This reference resolving process is used when generati
id1414G0I20HS | test para

id1414G0I20HS | test para

Figure 1-4: DITA-FMx Reference Manager dialog - conref insertion

If you double-click a conref or xref, the Reference Manager displays with the current reference selected. Choosing the Replace button inserts a new reference using the selected criteria in place of the old reference.

NOTE: *Replacing an existing reference removes any existing locally applied attribute data.*

Conrefs are inserted as a locked range of text (like a text inset) and are tagged with the “DITA-Conref” color. By default this color is defined as blue, but because it is defined in the template, you can change it to suit your needs.

Xrefs and links are also inserted as a locked range of text, but no coloring or formatting is applied other than that specified by your structure application (EDD or template). When an xref or link is created or modified, the type attribute is set to the name of the target element, the format attribute is set to “dita,” and the scope attribute is set to “local.”

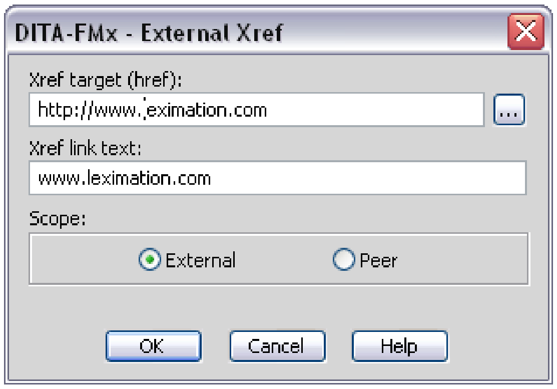
TIP: *If you’d like more of the title text visible in the Reference Manager, enable the “Use Wide Reference Manager” option in the DITA-FMx Options dialog.*

External Xref

Choose the External Xref button in the Reference Manager dialog to insert an <xref> or <link> that references files outside of the documentation set. Specify the target (the href attribute value) and the link text (the content of the <xref>

or <link> element), as well as the value for the scope attribute (“external” or “peer”).

For URLs, set the scope value to “external”. The format attribute is automatically set to “html” for references of this type.


DITA-FMx - External Xref

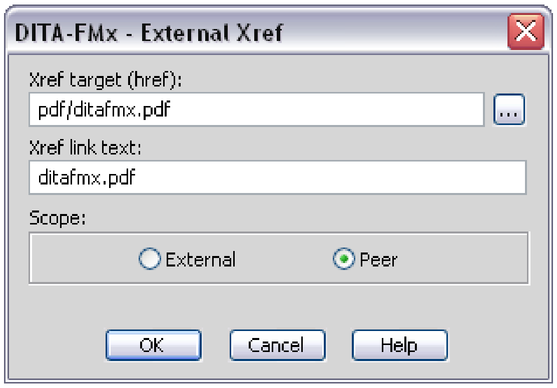
Xref target (href): http://www.leximation.com

Xref link text: www.leximation.com

Scope: [✓ External] Peer

Figure 1-5: DITA-FMx External Xref dialog box

If you are creating an <xref> or <link> to other types of non-DITA content use the browse button (“...”) to select the file or enter the path and file name in the href field. When using the browse button, the link text will be set to the target file name, you are free to change it to whatever is needed. The format attribute is set to the value of the file extension.


DITA-FMx - External Xref

Xref target (href): pdf/ditafmx.pdf

Xref link text: ditafmx.pdf

Scope: External [✓ Peer]

Figure 1-6: DITA-FMx External Xref dialog box

Keep in mind that it is your responsibility to make sure that the path in the href field is accurate for the “published” document. Just because you use the browse button to select a file that exists while authoring, doesn’t mean that this same path is valid for the published environment. Paths to external references (“external” or “peer”) are not validated during the DITA-FMx book build or other publishing processes.

Once an external reference has been created you can modify it by double-clicking the xref.

RELATED INFORMATION:

"INI-Only Settings" on page 51
"Update References" on page 21
"Insert Conref" on page 25
"Setting Up to Use Cross-References" on page 46

Filtering Content

Tips for filtering and conditionalizing content in topic files or maps.

DITA-FMx provides many ways to filter the content in your DITA files. The auto-conditionalizing options apply conditions to elements each time you open a file, and the **Apply Ditaval** command filters content by applying include or exclude actions (as defined in a ditaval file) based on attribute values. Ditaval filtering can be used on files while authoring (not recommended) and can be applied to the generated files in a book build. You can also achieve filtering when generating a book from a map by modifying the Book application's read-write rules.

Auto-Conditionalizing Elements

There are five options for applying conditions to elements on file open. The **Conditionalize prolog**, **Conditionalize comments**, and **Conditionalize required-cleanup** options apply the DITA-Prolog, DITA-Comment, and DITA-Cleanup conditions to the respective elements each time you open a topic file. The **Conditionalize data, data-about, and area** option applies the DITA-Data condition to <data>, <data-about>, and <area> elements when opening a topic file or DITA map file.The **Conditionalize topicmeta and book-meta** option applies the DITA-Topicmeta to the topicmeta element when opening a DITA map file. On file save, these conditions are stripped (all other conditions you may apply will honor the Conditional Text setting in the structure application definition).

The term "Conditionalize" does not necessarily mean to "hide," it just means that the associated conditional tagging will be applied to the elements. It is up to you to set the hide/show state as well as the color and style for these conditions (in the structure application template).

When any of these options are enabled, the plugin does a "show all" then a "hide" of the appropriate conditions when the XML or FM file is opened. If the file makes extensive use of conditions, this may result in the addition of blank pages at the end of the document. For XML files, this is not a problem since the blank pages are not saved, but if you're working on generated FM files, you may end up with extra blank pages that you can't get rid of (each time you save then reopen, the blank pages will reappear). If this is the case, you should disable the auto-conditionalizing options when doing final pagination on generated FM files.

Ditaval Filtering

To filter content in FM files based on a ditaval file, use the **Utilities** > **Apply Ditaval** command or enable the Apply Ditaval option in the book-build process. Before using this feature you must first create the ditaval file (using the **Ditaval Manager** or other means), then register it with the **Ditaval Manager**. Once the ditaval file has been registered with DITA-FMx, you can use it with the **Apply Ditaval** command. Detailed information about using this command is available in the Using the Apply Ditaval Command to Filter with Conditions topic.

NOTE: When using the Apply Ditaval option in the book-build process, you can reference a ditaval file directly in the book-build INI file without first registering it. If you do this, be sure to specify the full (or relative) path to the ditaval file (including the file extension). When referencing the registered ditaval in the book-build INI, just use the "name" you've given it in the Ditaval Manager.

As of DITA-FMx version 2.0.03, this command can be run in either of two modes, filtering *as conditions* or *by deletion*. Both modes result in similar results, but filtering *by deletion* will more closely match the filtering done by the DITA-OT.

While not a requirement, it is best to use this command only on generated FM files, not on your DITA XML files (as is done by the book-build process). If used on XML files, when using the filter *as conditions* option, the conditions are typically saved to the XML as processing instructions (PIs) and will persist in the state set the next time you open the file (this persistence is dependent on the Conditional Text setting in the associated structure application definition). If using the *by deletion* option, it will *delete* content from your files, which is typically not desired while authoring.

When you generate an FM book from a DITA map, any PIs created during authoring are filtered out during the import XSLT processing and are not included in the generated FM files. However, any conditions that have been applied to content that is the source of a conref, will appear in the generated FM files because the conref source is not passed through the import XSLT processing, but pulled in by the conref resolution process which happens after the initial topic file aggregation. This results in an FM file with some conditions that appear to have survived the conversion while other have not. This confusing situation can be avoided by not applying conditions to the XML files while authoring.

Regardless of whether you have applied conditions to the XML files or not, you can still run the **Apply Ditaval** command to apply conditions based on the properties in a ditaval file.

NOTE: In DITA-FMx, filtering can be performed with any attribute, not just the officially sanctioned attributes indicated in the DITA specification. (We don't necessarily encourage this, but there may be situations that warrant this use.) To enable this non-standard use with the filtering by deletion option, you must add the attribute name(s) to the INIOnly/FilteringAttributes setting in the ditafmx.ini file.

Filter by Attribute Build Expressions

If the filtering provided by the **Apply Ditaval** command doesn't result in the required filtering for your documents, you can make use of FrameMaker's **Filter by Attribute** build expressions when generating a book from a map.

NOTE: This feature is only available for FM10 and later (even though the command was available in FM8, the API was not available to support this until FM10).

To make use of this feature, you should start by developing (and extensively testing) the attribute build expression in a generated FM file (the result of the book-build process). When you've created the necessary build expression, import it into your Book template (or appropriate component templates) using **File** > **Import** > **Formats** and select the Filter by Attribute Settings option. In addition to the named build expression, your template must define the condition you plan to assign to the filtered content.

Use of this feature requires that you're using a book-build INI file. In the BookBuildOverrides section, add an entry named FilterByAttribute. Assign the build expression name and the condition name (separated with a greater-than symbol) as the value to this INI entry using the syntax: *BUILD-EXPRESSION>CONDITION-NAME*. For example, if your build expression is named "Print-Filter" and the condition is "FilteredContent", the INI entry would look like the following:

```
[BookBuildOverrides]
FilterByAttribute=Print-Filter>FilteredContent
...
```

Because the use of this feature is a replacement for ditaval filtering, if this entry is used in the book-build INI file, any specified ditaval filtering is ignored for content filtering. When the FilterByAttribute feature is in use, ditaval filtering is still used for the filtering of key definitions, so it is important to maintain both.

In the future, DITA-FMx may provide a mapping from ditaval to attribute build expressions, but for now this is left to the user to develop.

Filtering with the Read-Write Rules File

If there are certain elements that you want to exclude from your generated FM book files, use the "drop" rule. For example, if you want to exclude the shortdesc element from your generated files, add the following rule to the Book application's rules file:

```
element "shortdesc" drop;
```

This results in the shortdesc element being removed from the generated FM files while it remains in the source topic files.

RELATED INFORMATION:

"DITA Options" on page 33
"Ditaval Manager" on page 18
"Apply Ditaval" on page 4
"Using the Apply Ditaval Command to Filter with Conditions" on page 25

Using the Apply Ditaval Command to Filter with Conditions

Steps for using a ditaval file to apply conditional filtering to topics.

PREREQUISITES:

When run interactively, the ditaval file must be registered with DITA-FMx. To create a new ditaval file or register an existing ditaval file, use the **Ditaval Manager** command.

TASK

1. Open the document or book to apply the conditions.
2. Run the **DITA-FMx > Utilities > Apply Ditaval** command.
3. In the Apply Ditaval dialog, select the **Filter with Conditions** option, then select the ditaval file to use from the list box.
4. For each "action" type defined in the ditaval file, set up the options in the appropriate "action=" section.

   ADDITIONAL INFORMATION: For example, if your ditaval file contains two prop elements where each action attribute is set to "exclude", you need to decide if the condition to apply to the matching elements should be a specific name (one name for all “excluded” elements) or if you want it to apply a unique name for each prop element. If you’re OK with using a single condition name, just select the radio button under Condition Name and enter that name in the text box. If you’d like a unique name for each prop element, select the “<attribute>=<value>” radio button. This will make two conditions (one for each prop element), and apply each to the corresponding elements.

Select the visibility option for the condition. You can always change the visibility after applying the conditions using the standard FrameMaker condition management commands.

5. Choose the OK button to apply the conditions.

RELATED INFORMATION:

"Apply Ditaval" on page 4

Working with Images

Information and tips regarding the image handling features.

To insert an image, just insert the <image> or <fig> element from the Element Catalog. The Insert Image dialog lets you select the image folder location and the image file name. You can also set the @alignment, @placement, and DPI values. If the “fmdpi” feature is enabled, the DPI value will be assigned to that property, otherwise it’ll be used to set the @height and @width attribute values.

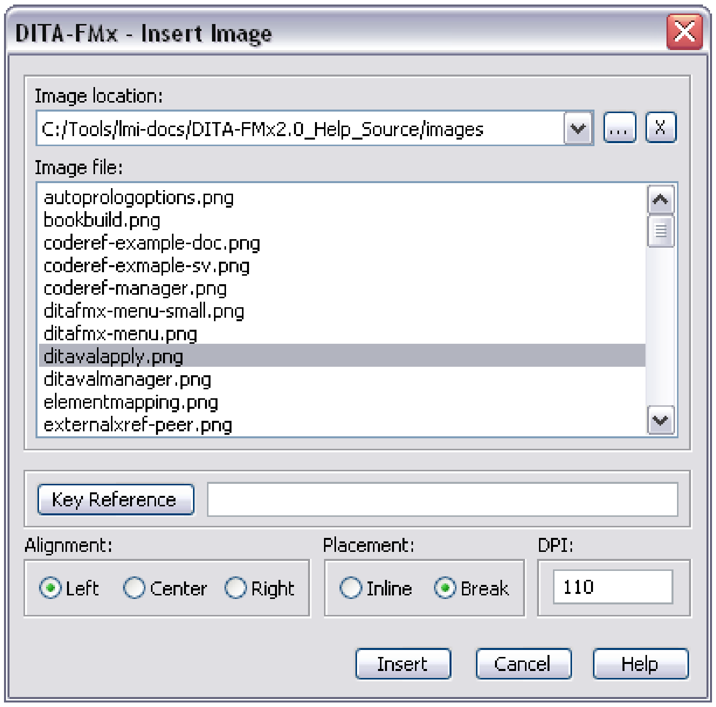


Image location: C:/Tools/lmi-docs/DITA-FMx2.0_Help_Source/images

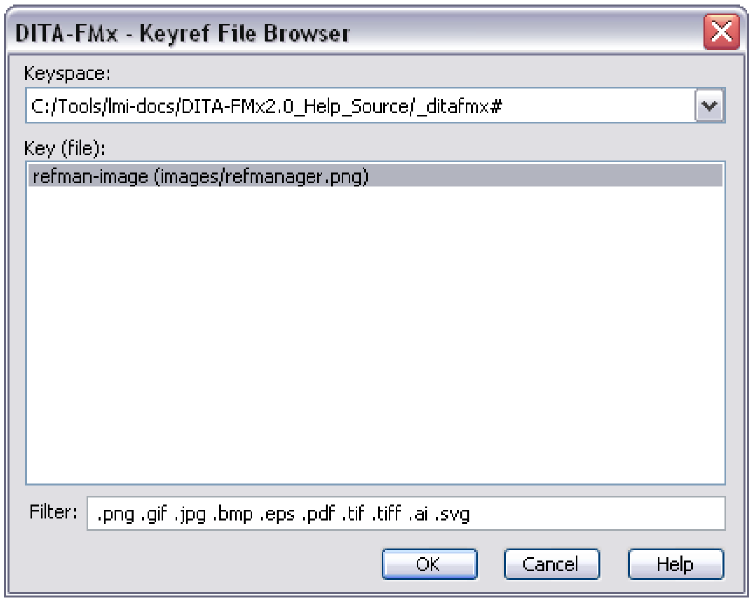
Image file:
autoprologoptions.png
bookbuild.png
coderef-example-doc.png
coderef-exmaple-sv.png
coderef-manager.png
ditafmx-menu-small.png
ditafmx-menu.png
ditavalapply.png
ditavalmanager.png
elementmapping.png
externalxref-peer.png

Alignment: [✓ Left] Center Right
Placement: Inline [✓ Break]
DPI: 110

Figure 1-7: Insert Image dialog

Use the "..." (browse) button to add one or more image location paths to the list. The "X" button deletes the current path from the list.

Alternatively (or in addition to selecting the file name), you can specify a key reference to identify the image file. Choose the Key Reference button to display the Keyref File Browser dialog. Selecting the keyspace lists all of the keys in that keyspace that reference files which match the specified file name filter.



Keyspace: C:/Tools/lmi-docs/DITA-FMx2.0_Help_Source/_ditafmx#

Key (file): refman-image (images/refmanager.png)

Filter: .png .gif .jpg .bmp .eps .pdf .tif .tiff .ai .svg

Figure 1-8: Keyref File Browser dialog

When an image is added to a document, the anchored frame is "shrink-wrapped" to the size of the image. The relative path to the image file is added to the @href attribute and the height/width in pixels is added to the @height and @width attributes. The initial value of the @placement attribute matches that specified as the default in the EDD, unless the image is auto-inserted as the result of inserting a <fig> element in which case the @placement attribute is set to "break." For images where the @placement attribute value is "break," the initial alignment is set to match that of the default value of the @align attribute as defined in the EDD; no alignment value is set for "inline" images.

You can change the alignment after inserting the image by editing the value of the @align attribute or by selecting an alignment option in the Anchored Frame dialog (**Special > Anchored Frame**). The @height and @width cannot be set through the attribute value; they can only be changed by re-importing the image using a different scaling, or by changing the properties through the Object Properties dialog. The @href attribute also cannot be changed through the attribute value.

If you do not want any height/width values associated with an image, deleting those attributes in the XML or Frame document will result in the image element being written with no values for @height and @width which uses the "native" (default) size of the image. This is often desirable for output being generated as HTML.

If you want to use the native size of raster images for output generated through the Open Toolkit but want to use a specific DPI for output generated from FrameMaker, add an "fmdpi:*DPI*" attribute value to the image/@outputclass attribute, where *DPI* is the DPI value. Adding this value to the @outputclass attribute allows the DPI setting to round trip from XML to Frame and it will only be used by FrameMaker; other output processes will see no @height and width @values and will use the native image size. Adding this value to the @outputclass attribute deletes any value set to the @height or @width attributes.

If you want to use the fmdpi feature for all raster images, enable the "Use fmdpi for New Images" setting in the Authoring Options dialog. This will set the fmdpi value to the DPI value you select when importing the image. The fmdpi value is only set for raster images; this feature is not available for vector images.

NOTE: *If your images sizes are not round-tripping properly, your read/write rules file probably needs to be updated. Refer to the image topic for the proper rules, especially with regard to the height and width attribute definitions.*

Baseline Offset of Inline Images

By default, the baseline offset for inline images is set to 2 pts (this shifts the image 2 pts below the baseline of the surrounding text). If you want to set this property to a different value, change the value of the Baseline Offset option in the Authoring Options dialog. This value is only applied to inline images. Setting the “baseline offset” property in the read/write rules file has no effect.

Special Handling of Alternative Text (image/alt)

Content of an image/alt element is stored in the anchored frame’s “object attribute” while open in FrameMaker. To edit the alt text, select the anchored frame, then choose **Graphics > Object Properties**, then Object Attributes. The <alt> value is stored in the “Alternate” field.

Because a graphic object cannot have child elements this content is stored in an attribute, any attributes on the <alt> element or child elements are discarded when the file is opened in FrameMaker.

Related information:

"image" on page 35
"INI-Only Settings" on page 51
"Authoring Options" on page 47

Graphic Overlay Objects

Working with text callouts and simple graphic objects placed over a referenced image.

DITA-FMx supports the use of callouts and other graphic overlay objects in DITA topics. Although this is not specifically supported in DITA, the data to round-trip these objects is stored in nested <data> elements immediately following the <image> element within a <fig>. This support is only available for images that are child elements of a <fig> element.

DITA-FMx supports the following types of graphic objects as overlay objects within an anchored frame:

- Line
- Arc
- Polyline
- Rectangle
- Rounded Rectangle
- Oval
- Polygon

- Text Line
- Text Frame (limited support)

Note that a Text Frame can only contain a single paragraph and the only properties that are saved are horizontal alignment and paragraph style. Other types of nested frames are not currently supported. Additionally, the object grouping mechanism is not supported.

The Text Line object uses the character style applied to the first character, and applies that style to all characters in the object. The Text Frame object stores the paragraph style name that is applied when it is created. Both of these styles, the character style for Text Line objects and the paragraph style for Text Frame objects, should be predefined in the Topic and Book templates so they will be formatted properly in the final output.

If you plan to edit callouts as XML (for localization or other purposes) you should not use the Text Line object. The dimensions of the text are stored in the data elements and will not update when you change the actual text. The result will be that the callout remains the same size with the text stretched or compressed to fit into the same area. If you plan to edit the callout text as XML, you should use the Text Frame object which defines the bounds of a region in which the text will flow.

The data stored in the nested <data> elements is very FrameMaker-specific, but can be translated into other formats such as SVG with a little effort. If your <image> element is within a <fig> element and contains graphic overlay objects, a <data> element will be inserted immediately following the <image> element. This <data> element will have the @name attribute set to “fm:imagedata”. Within this <data> element will be a child <data> element for each of the graphic overlay objects, and each will have a @name attribute set to a value appropriate for the object type (e.g. “fm:textline” for a TextLine object and “fm:polyline” for a Polyline object).

NOTE: *Prior to v.2.0.06 graphic overlay objects used the @datatype attribute to store the object type value (fm:imagedata, fm:textline, etc.). This was switched to use the @name attribute for LwDITA support. The migration from @datatype to @name will happen when you open and save a DITA file. If you want to continue using the @datatype attribute, set the INIOnly/UseDatatype parameter to “1”.*

When written to XML, all of these <data> elements will be empty elements (just specifying @name and @value attribute values) except for the text of a Text Frame object. Because this content is likely to be modified for localization purposes, it is written to XML as content of the <data> element. On import to FrameMaker, this is converted to a data/@value.

Each of the child <data> elements will have the @value attribute set to a tilde-delimited string that specifies the property names and values. For some properties the associated value may be a very large numeric value, this is a

FrameMaker “MetricT” value (this has nothing to do with the metric measurement system). When used for linear measurement, value of 65536 is equal to a point (1/72 inch).

This feature can be disabled in the Authoring Options dialog.

Support for Indented Images

Images are typically aligned with the text margins, but you can enable the ability for images to align with the container paragraph.

The image/@placement attribute controls the anchoring position of images. By default, when inserting an <image> without a <fig> container, the @placement attribute will be set to “inline” (since this would typically be used for inline images). When inserting an <image> within a <fig> (inserting the <fig> typically automatically inserts the <image>), the @placement attribute will be set to “break”. You are free to change the @placement value as needed for situations that require a different setting. When @placement is set to “inline” the anchoring position (**Special > Anchored Frame**) is set to “At Insertion Point”, and when @placement is set to “break” the anchoring position is set to “Below Current Line” (these settings cannot be changed).

When @placement is “break” and @align is “left”, the image will be forced to the left margin (page margin or side head margin). If you are using side heads this is typically the formatting that’s needed, but if you’re not using side heads or when you have an image in indented text (like a list), you will probably want the image to be indented to match the text block.

If you make use of the “Move Figure Title” book-build option, you will find that setting the @placement of an <image> within a <fig> to “inline” will probably achieve the desired result by allowing the <image> to align with the containing paragraph. To automate this processing, DITA-FMx provides the “BreakToInline” parameter in the BookBuildOverrides section of the book-build INI file (*ditafmx-bookbuild.ini*). Setting this parameter to “1” will change all of the `placement='break'` images to `placement='inline'` that are in a paragraph whose first indent is greater than zero. This setting must be made manually in the book-build INI file, and may not work well in all situations, so it’s best to experiment a bit and test to make sure it is providing the proper results.

In order for this to work properly, the “Fixed” option of the “Line Spacing” property of the fig.anchor paragraph (or the style assigned to a figure anchor) style must be deselected. Also, the MoveFigTitles book-build INI parameter must be set to “1”. When testing this, be sure to watch the report in the console log to verify that you see the following message:

```
Using BookBuild Override: BreakToInline=1
```

Image Formatting Options

Default DITA attributes provide for very basic formatting of images. DITA-FMx supports the use of special outputclass attribute values to leverage FrameMaker's enhanced image layout options.

To apply special layout features to an image, add one or more of the following values to the image element's @outputclass attribute. Note that there is not a one-to-one correlation between these values and the interface options, so some experimentation will likely be needed to get exactly what you want.

- fm:pos-below - Below Current Line
- fm:pos-top - At Top of Column
- fm:pos-bottom - At Bottom of Column
- fm:pos-inline - At Insertion Point
- fm:pos-runin - Run into Paragraph
- fm:pos-col-left -
- fm:pos-col-right -
- fm:pos-col-near -
- fm:pos-col-far -
- fm:pos-col-inside -
- fm:pos-col-outside - Outside Column
- fm:pos-frame-left -
- fm:pos-frame-right -
- fm:pos-frame-near -
- fm:pos-frame-far -
- fm:pos-frame-inside -
- fm:pos-frame-outside - Outside Text Frame
- fm:align-left - Align left
- fm:align-center - Align center
- fm:align-right - Align right
- fm:align-inside - Align inside
- fm:align-outside - Align outside
- fm:aframe-cropped - Anchored frame, cropped

- fm:aframe-floating - Anchored frame, floating
- fm:baseline-<UNITVAL> - Above Baseline
- fm:gap-<UNITVAL> - Gap

NOTE: *<UNITVAL> = 60pt, 60%, etc (no space before or within <UNITVAL>)*

Working with Tables

DITA provides tables for many situations. Learn how to make them look the way you want.

DITA-FMx provides a number of methods for controlling the appearance of tables. You can't just apply ad-hoc formatting to a table and expect it to "stick", but with a little planning and setup, you should be able to achieve most of the formatting that you want.

General table formatting is accomplished by assigning a FrameMaker table format. Custom ruling and shading can also be achieved by setting up the template's reference page with representative examples of the custom formatting. By default, tables are rendered to fill the full width of the text column, but can be adjusted. Rotated table cells will round-trip without any special effort. If you make use of specialized elements based on the simpletable element, you'll need to provide information about those structures in order for them to operate properly.

RELATED INFORMATION:

"Element Mapping" on page 55

Table Formatting

Named table formats allow you to control the basic layout and appearance of each table.

When inserting a table of any type, you are prompted to select from one of the available table formats (defined in the structure application template). You can modify the definition of these table formats or add new named formats to the template. Be sure to update both the "Topic" and "Book" structure application templates so the same table formats exist in both places.

The assigned table format name defines the basic formatting of the table. This name is applied to the outputclass attribute of the "table" object element (tgroup for table elements).

TIP: *Previous structure applications included a fixed list of table formats in the element definition, which was cumbersome to maintain. The latest DITA-FMx structure application uses a modified EDD structure that allows for the value of outputclass to be used as the Initial Table Format. This means that when adding new formats, you just need to update the template, and not the EDD. If you're using a custom EDD, be sure to import the EDD from the DITA-FMx EDD into your EDD.*

RELATED INFORMATION:

"Working with Tables" on page 33

Indented Tables

It may be desirable to indent tables within lists.

Unfortunately, FrameMaker does not provide any way to control the indenting of tables other than by setting a left indent value for all tables of a particular format. While you could have a special format for tables in lists, this may quickly become unreasonable to manage.

DITA-FMx provides the Indent Table to Parent Element option (in the Authoring Options dialog), to automatically set the left indent for tables, based on the Left Indent property of the parent element's paragraph tag. When a table is used in a standard list, the parent element will be the <li> element.

The indent is applied on file save (as well as on file open). After inserting a table in a list, save the document and the table will be indented to match the parent.

The indenting of <simpletable> elements is controlled by the @expanse attribute values:

- **expanse='textline'** - always indents to the parent element, regardless of the setting of the indent option.
- **expanse='column'** - never indents; always forces the table to the column width.
- **expanse='page'** - never indents; always forces the table to the page frame width.
- **expanse='spread'** - never indents; always forces the table to the page frame width.

NOTE: *For book builds, if your page layout uses varying column or page frame widths, you may need to use the BookBuildOverrides/FixTableWidths=1 setting.*

RELATED INFORMATION:

"Working with Tables" on page 33
"Authoring Options" on page 47

Custom Ruling and Shading

Applying ruling and shading to specific table rows and cells requires additional forethought and setup.

DITA-FMx allows you to apply custom ruling and shading to table rows or cells based on the @outputclass attribute. Because there is no option to control this formatting through EDD context rules, you must set up a representative table on an “fmx-tableformat” reference page in the structure application templates. A sample page has been added in the default DITA-FMx Topic and Book templates.

To enable this feature, you must enable the Apply Custom Ruling and Shading option in the Authoring Options dialog.

On this reference page (must be created if it doesn’t exist), create a text frame and add a structure flow that contains the table types for which you want to provide formatting. Use the **Table > Row Format** or **Table > Custom Ruling and Shading** commands to assign the desired formatting to a row or cell. Then set the outputclass attribute of that formatted row or cell to the value you want to use to identify that formatting. Save the template. Be sure to make similar modifications to both the Topic and Book templates.

After setting up the template, apply the appropriate name to the @outputclass attribute of row or cell elements in your DITA topic files. When you save the file, the formatting is transferred from the object on the reference page to the corresponding object in the topic.

Keep in mind that the formatting is only applied to matching row or cell elements. Don’t set the @outputclass on other table objects (<thead>, <tbody>, etc.). For example the formatting of an <entry> element on the reference page with an @outputclass of “darkblue” will not be applied to an <stentry> element with an @outputclass of “darkblue,” only to another <entry> element.

Also, note that these properties are only applied by row or cell. The properties applied by row are: row min/max height, row start position, and row keep w/ next/previous. The properties applied by cell are ruling and shading (coloring). In order to apply both row and cell properties, you must define both types of schemes in your reference page table. Developing very complex ruling, shading, etc. layouts can be very difficult, so start with a simple case and slowly add more properties until you understand how this is done.

RELATED INFORMATION:

"Working with Tables" on page 33
"Authoring Options" on page 47

Controlling Table Widths

Tables will typically fill the available text column width, but there are options if you need specific control over table widths.

If you are using the default DITA-FMx structure applications, all tables, regardless of their type, are set to use proportional widths. This means that the table will expand to fill the width of the current text column and all cells within the table will proportionally fill the table based on the widths you assign.

If you want to be able to specify absolute widths for tables, you need to comment out or delete the `writer use proportional widths` rule in the read/write rules files (*topic_1.2.rules.txt* and *book_1.2.rules.txt*). After commenting this line out it will look like the following:

```
/* writer use proportional widths; */
```

NOTE: *This setting affects all tables of all types that use the structure application associated with the rule file. You cannot have some tables that are proportional and some that are absolute.*

You will also need to disable the **Force Tables Wide** option in the **Authoring Options** dialog.

If you'd like a specific table to fill the width of the text frame (overriding the margins or indents), set the <table> element's @pgwide attribute to 1. This attribute is only available for tables that inherit from the base <table> element.

An alternate option for defining table widths that vary is through the use of the **Preserve Table Widths** option in the **Authoring Options** dialog. This should be used with **Force Tables Wide** enabled, and will preserve the width of tables to the nearest 5%. A relative table width value is stored in the table's @pgwide attribute (this value is updated on file save).

Use the **Preserve Table Widths** option only if the "writer use proportional widths" rule is enabled, if disabled, this feature is not needed. Because of the 5% adjustment on save, column widths will not remain exactly as set (they will be close though).

IMPORTANT: *If you're using the **Preserve Table Widths** option, problems can occur if the text flow width in the Topic and Book templates are not the same, because the table width is preserved as a percentage of the text flow width. The default DITA-FMx Topic and Book templates do not have matching width text flows. Edit the Reference pages in Topic template to set the text flow width to match that of the Book.*

RELATED INFORMATION:

"Working with Tables" on page 33
"Authoring Options" on page 47

Rotated Table Cells

FrameMaker allows you to easily rotate table cells, and this will remain rotated for PDF publishing.

If the Save Cell Rotation option is enabled in the Authoring Options dialog, table cell rotation will be preserved. This is typically done to specify long table column headings, but can be done on any table cell. Because this feature is not specifically supported by DITA, it's not likely that this rotation will be applied to output types other than those generated through FrameMaker.

To rotate a cell, select the entire cell, then choose **Graphics** > **Rotate** and specify the rotation angle. Save, close and reopen the file in FrameMaker, and the rotation will be applied.

This rotation information is stored as a DITA data element (the first child of the entry element). This data element has a @type attribute of "fmx-rotated" and a @value attribute that specifies the rotation angle. Because this is valid DITA markup, there should be no adverse reaction in other editors or processing tools. In fact, it would be possible for other processors to key off of this element to apply a cell rotation for other output types.

Related information:

"Working with Tables" on page 33
"Authoring Options" on page 47

Simpletable Specializations

A number of the DITA table types are specializations of the simpletable element, if you create additional specializations, they need to be identified to open properly in DITA-FMx.

When simpletable-based elements are encountered during the import process, FrameMaker needs to be able to count the number of columns in each table. This information is typically stored in attributes within the table element, but the DITA specification does not provide this type of attribute for simpletable-based tables.

The DITA-FMx Options dialog provides access to the Element Mapping dialog which defines the structure of the default simpletable-based tables, and allows you to define the structure of any specializations you create.

Related information:

"Working with Tables" on page 33

Working with Hazard Statements

The hazardstatement element model can be very useful for certain documentation types, but authoring and publishing in a consistent manner can be tricky.

With the release of DITA-FMx 2.0.06 you can now make effective use of hazardstatement elements in your topic files. There are a number of options to control the processing while authoring as well as various options and setup requirements for publishing.

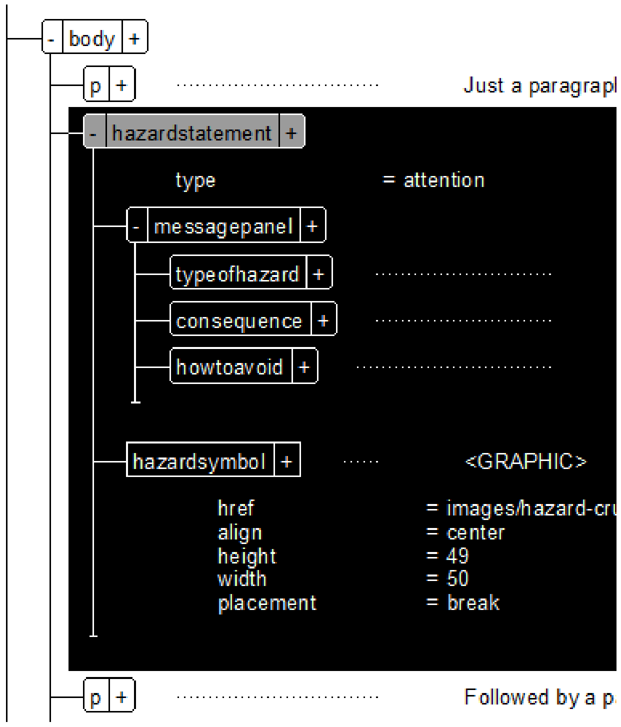
RELATED INFORMATION:

"Authoring Hazard Statements" on page 39
"Hazard Statement Publishing Options" on page 40

Authoring Hazard Statements

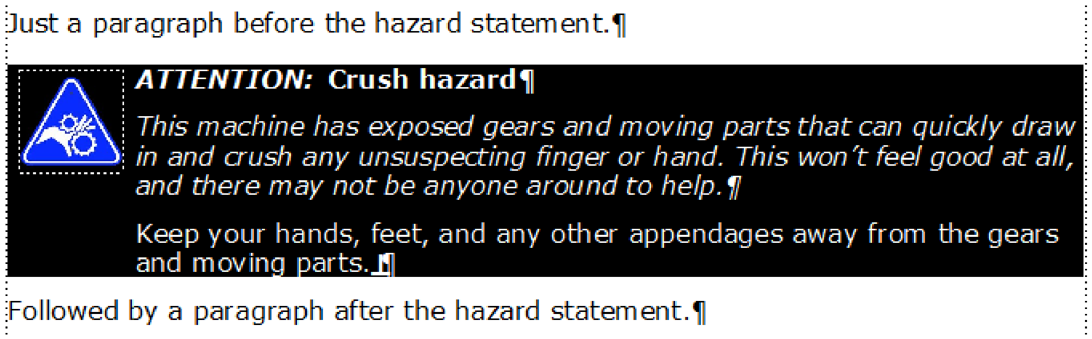
In order to make them easier to visualize, DITA-FMx performs minor adjustments to <hazardstatement> elements when authoring.

The DITA model places the <hazardsymbol> element at the end of the <hazardstatement> model. This can make for a somewhat odd-looking layout when authoring. The default DITA-FMx publishing process wraps this model in a special hazard table to make it all look right, but that can't be easily done for authoring.



As of DITA-FMx 2.0.06, special processing of <hazardstatement> elements while authoring, is done if the Process Hazardstatement Elements option is enabled in the Authoring Options dialog.

The default DITA-FMx Topic application, indents the <hazardstatement> elements which allows the <hazardsymbol> image to be moved into the space to the left of the hazard information. The Max Symbol Width setting in the Authoring Options dialog controls the maximum image width; be sure to set this to a value that fits in the available space.



Just a paragraph before the hazard statement.¶

ATTENTION: **Crush hazard**¶

This machine has exposed gears and moving parts that can quickly draw in and crush any unsuspecting finger or hand. This won't feel good at all, and there may not be anyone around to help.¶

Keep your hands, feet, and any other appendages away from the gears and moving parts.¶

Followed by a paragraph after the hazard statement.¶

The hazard type label ("ATTENTION:" in the image above) is controlled by the autonum prefix of the paragraph tag assigned to the <typeofhazard> element. This paragraph tag is set based on the @type attribute of the <hazardstatement> element. The position of the <hazardsymbol> image is set when the image is inserted. It's also updated on file save, so it may not look quite right after adding more content to the hazard statement until you save. If multiple <hazardsymbol> elements are included, only the first is placed in the indented space; the rest are added after the last <messagepanel> element.

Setting the @align attribute of the <hazardsymbol> element controls the horizontal placement within the indented area as well as in the published output.

NOTE: *A prefix/suffix on the <hazardstatement> element is not supported by the default processing.*

RELATED INFORMATION:

"Hazard Statement Publishing Options" on page 40
"Authoring Options" on page 47
"INI-Only Settings" on page 51

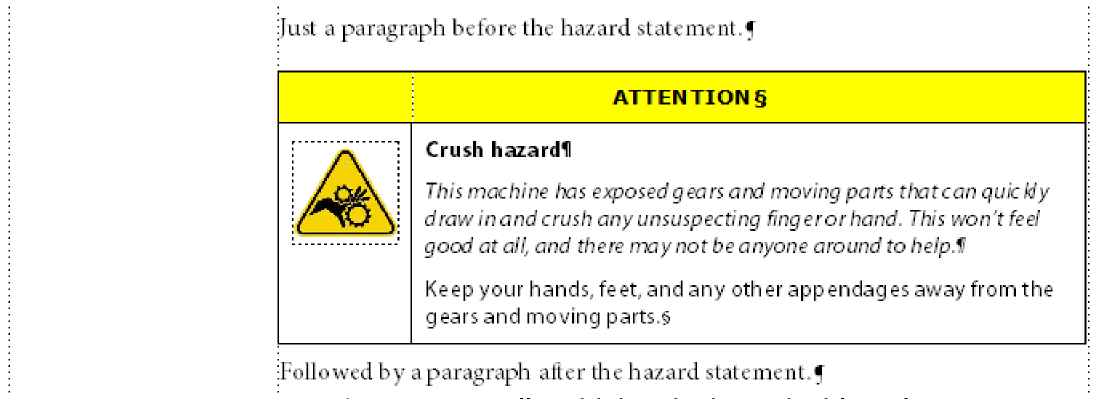
Hazard Statement Publishing Options

To render <hazardstatement> elements for publishing, the default DITA elements are wrapped in a table structure made up of a number of special <fm-> elements.*

NOTE: *The <fm-*> elements mentioned here are described in DITA-FMx Specific Book Application Elements.*

The hazard table wrapping of <hazardstatement> elements is performed by default if the special <fm-*> elements exist in the Book structure application. To prevent this process from running, in the bookbuild INI, set BookBuildOverrides/HazardToTable=0.

DITA-FMx supports two styles of hazard tables. One style has a full-width header and a row below with two cells, one for the hazard symbol and the other for the hazard information (<messagepanel> structure). The other style has a partial-width header where the hazard symbol is in the first column, and the header and content in the second column (this is the default style in the DITA-FMx Book templates).



Just a paragraph before the hazard statement.¶

ATTENTION§

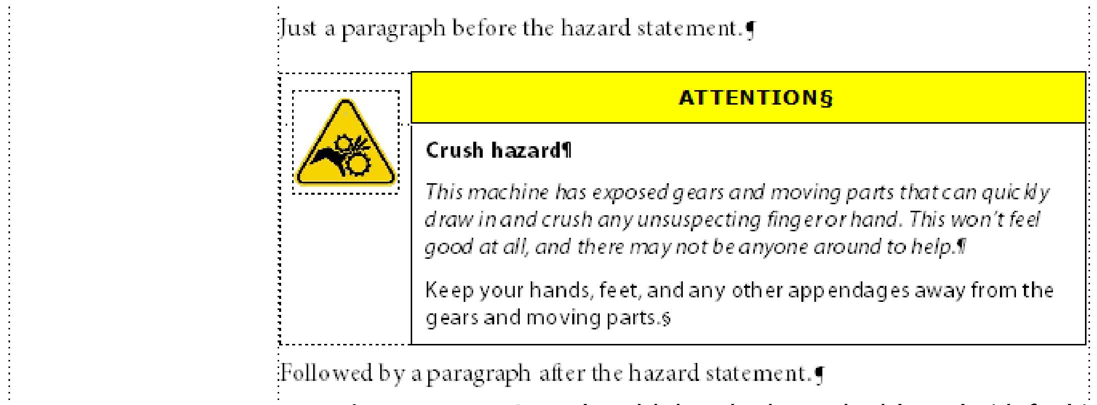
Crush hazard¶

This machine has exposed gears and moving parts that can quickly draw in and crush any unsuspecting finger or hand. This won't feel good at all, and there may not be anyone around to help.¶

Keep your hands, feet, and any other appendages away from the gears and moving parts.§

Followed by a paragraph after the hazard statement.¶

Figure 1-9: Full-width header hazard table style



Just a paragraph before the hazard statement.¶

ATTENTION§

Crush hazard¶

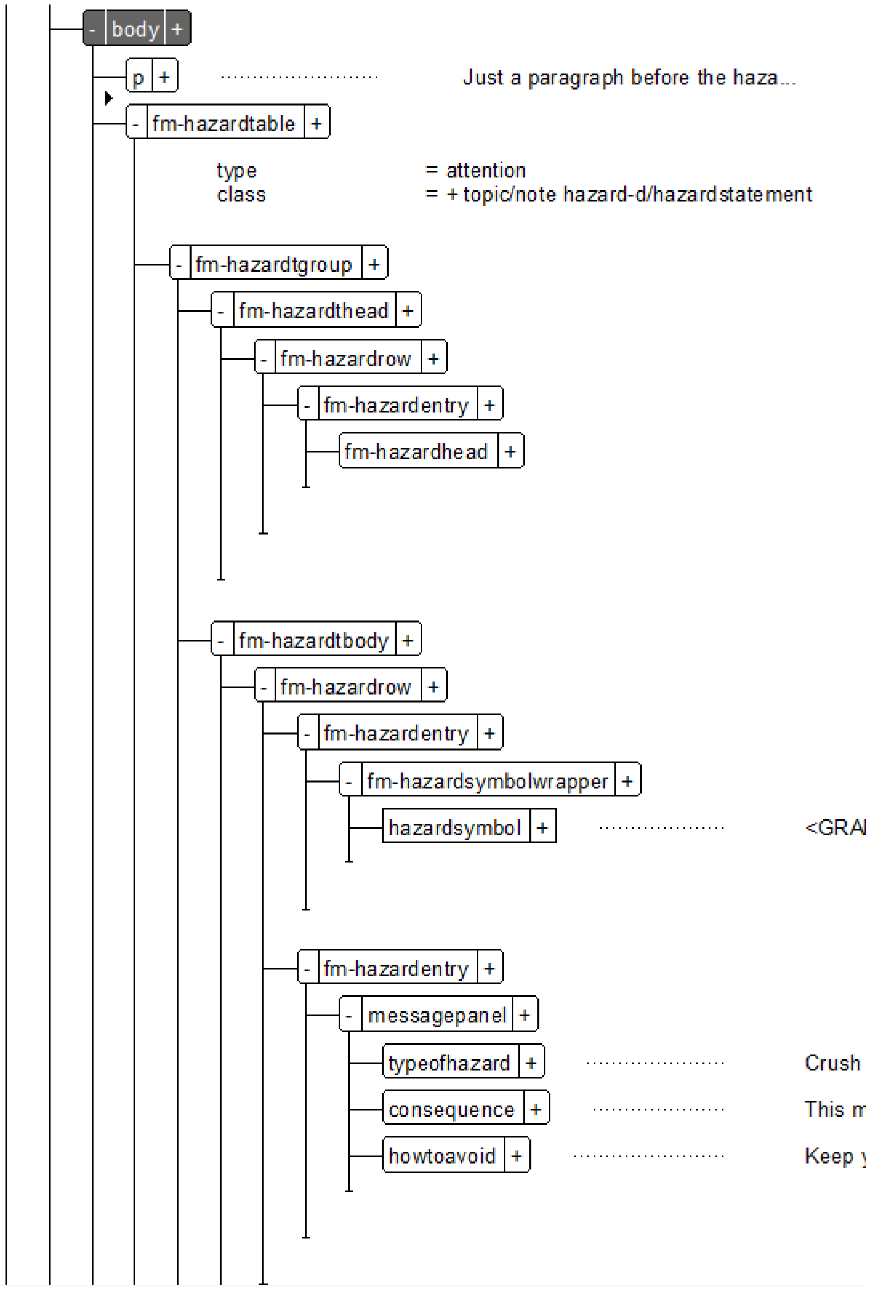
This machine has exposed gears and moving parts that can quickly draw in and crush any unsuspecting finger or hand. This won't feel good at all, and there may not be anyone around to help.¶

Keep your hands, feet, and any other appendages away from the gears and moving parts.§

Followed by a paragraph after the hazard statement.¶

Figure 1-10: Partial-width header hazard table style (default)

During the hazard-to-table processing, each <hazardstatement> element is replaced with an <fm-hazardtable> element and the attributes are transfered from one to the other. Within the <fm-hazardtable> a <fm-hazardtgroup> element is inserted which inserts a “Hazard” table object. The default properties of this table controls the appearance of the resulting hazard tables. To create a full-width header table, the default layout should have 1 header row and 1 body row, with 2 columns. For a partial-width header table, set up the default table layout with no header rows, 2 body rows and 2 columns.



type = attention
class = + topic/note hazard-d/hazardstatement

Figure 1-11: Structural model after hazard-to-table processing

The header background coloring is controlled by a custom color definition named based on the hazardstatement/@type attribute. Create a custom color for each @type value you plan to use. These colors should use the naming convention of “hazard-*TYPE*” (where “*TYPE*” is the value of the @type attribute). Hazard statements without a @type attribute value will use a color named “hazard.”

The formatting and text of the header label is also tied to the @type attribute. A paragraph tag is applied to the <fm-hazardhead> element using the naming convention of “hazard.head.*TYPE*” (and “hazard.head” for no @type value). This paragraph tag defines an autonum prefix for each label for the header text.

The <hazardsymbol> element is inserted into the first column and wrapped in a <fm-hazardsymbolwrapper> element. This wrapper element is used even when there is no <hazardsymbol> element to allow custom formatting if needed. For the partial-width header table, the top, left, and bottom rulings are set to “None” for the first column, effectively making the symbol separate from the content table.

The maximum width of hazardsymbol images is controlled by the Max Symbol Width setting in the Authoring Options dialog. To override this for a book build, use the BookBuildOverrides/HazardSymbolMaxWidth parameter.

The <messagepanel> element (and children) are inserted into the second cell in the second row. Additional <messagepanel> elements are inserted into extra rows in the table. Use typical EDD/template formatting techniques for the <messagepanel> elements.

If you are using the full-width header tables and want to create separate table formats for each hazard type, you can use the header background color defined in the default format rather than creating custom color definitions. To do this, in the bookbuild INI set BookBuildOverrides/UseColorFromTable=1.

The default DITA-FMx Book application and component templates all have the necessary settings, objects, and element definitions to render the partial-width header hazard tables (this format appears to be the most “standard”). If you’d like to test the full-width header style, locate the *tpl~topicref-HAZALT.fm* file in the Book application’s *component-templates* folder. This file contains the styles and objects that will create a full-width header style hazard table. If you’ve enabled the “Component Templates” book-build option, and your component map element types are “topicrefs” you can swap this file for the default topicref component template. Or, if needed, rename it to be a chapter component template. To set up other templates, you’ll need to copy and paste the Hazard table into that template as well as transfer the color definitions and paragraph definitions.

RELATED INFORMATION:

"Authoring Hazard Statements" on page 39
"Book Build Settings" on page 57

Working with Indexterms

Special issues regarding the creation of index entries (indexterm and fm-indexterm elements).

DITA-FMx supports the use of the DITA indexing elements in FrameMaker by mapping them to the appropriate Index marker syntax, if the Indexterm to fm-indexterm option is enabled (in the Options dialog). When the Indexterm to fm-indexterm option is enabled, you must use the <fm-indexterm> element rather than <indexterm>.

In general, the process of creating an index entry works as it does in unstructured FrameMaker. Inserting an <fm-indexterm> element displays the **Insert Marker** dialog where you type the index entry using the standard FrameMaker index syntax. On file save, this marker syntax is converted into the proper DITA <indexterm> structure. When opened again in FrameMaker, this is converted to the FrameMaker marker syntax as an <fm-indexterm> element. When creating an index entry, feel free to enter multiple entries in a single Index marker (separated by semicolons and using the standard FrameMaker marker syntax). When you reopen the file in FrameMaker, you'll see multiple <fm-indexterm> elements, because the single Index marker is converted into separate <indexterm> elements.

NOTE: *For FrameMaker language versions that use a translated name for the "Index" marker (such as Japanese), you must edit the ditafmx.ini file and set the IndexOptions/IndexMarkerType parameter to the name of the Index marker on your system. In doing so, you may need to save the INI file as UTF-8 to prevent the characters from being corrupted.*

In order to properly round-trip the "see" and "see-also" index entries, you need to include specific character styles within the marker syntax. For example to make a "see" entry, you would use the following syntax that uses the "ix-see" character style:

```
<$nopage>ONE:TWO. <ix-see>See</> SOMEWHERE, ELSE
```

To make a "see-also" entry, you would use the following syntax that uses the "ix-seealso" character style:

```
ONE:TWO;<$nopage>ONE:TWO:<ix-seealso>See also</>
SOMEWHERE, ELSE
```

IMPORTANT: *You should not include the "forced sort" information in a "see" or "see-also" entry; that will be added based on the Forced Sort Value you enter in the DITA-FMx Index Options dialog. For example, if your Forced Sort Value is "zzz" the Index marker syntax of the sample entry above would be* `ONE:TWO;<$nopage>ONE:TWO:<ix-seealso>See also</> SOMEWHERE, ELSE[ONE:TWO:zzz]`*. The forced sort information is added* *when you save the file; if you manually enter forced soft information it will likely conflict with the automatically added value.*

These character style names are defined in the **Index Options** dialog, so you can change them as needed. However, if you do change them, be sure to create the appropriate styles in the Topic and Book templates ("ix-see" and "ix-seealso" are the defaults).

The only way for DITA-FMx to know when you want an index entry to be a "see" or "see-also" is by entering the marker text in a specific way and to use these predefined character styles. If you enter the marker text differently, it may not properly convert into the DITA <indexterm> structure. Also note that after being saved to DITA and re-opened in FrameMaker, the marker text may vary slightly from what you entered (just punctuation and structure, not content). The **Index Options** dialog provides a number of options that allow you to define the way the <indexterm> content converts into the FrameMaker marker syntax.

DITA-FMx supports index page ranges through the <indexterm> @start and @end attributes. To create an index page range, insert an <fm-indexterm> element at the start of the range and set the @start attribute to a unique value. Then insert another <fm-indexterm> at the end of the range and set the @end attribute value to the same value as the @start attribute. There is no need to enter the "<$startrange>" and "<$endrange>" FrameMaker index marker syntax; this will be added automatically when the file is reopened. The actual text of the "end" marker does not need to match that of the "start" marker, as it will be synced with the "start" marker (the end marker does need to have content, it cannot be left empty). If the "end" marker is located in a different file from that of the "start" marker, the end marker's content will be "EMPTY", but this value will be properly synced when the files are aggregated into chapter files at book-build time. Note that not all publishing processes will support index page ranges.

The DITA <indexterm> structure allows for elements that have no direct parallel in FrameMaker. If you need to work with very complex <indexterm> structures, you should disable the Indexterm to fm-indexterm conversion option in the **Options** dialog. This will give you access to the full array of elements provided by DITA and you'll be able to insert an <indexterm> element (as a container object, not a marker) and enter the specific DITA elements that you need. This is required if you want to use any non-index elements as children of an <indexterm>.

RELATED INFORMATION:

"Index Options" on page 52

Working with Maps

General information on creating maps and bookmaps, and how they work in FrameMaker.

The default Map structure application (DITA-FMx-Map-1.2) allows for creation and editing of both DITA map and bookmap files. When saved to disk, the resulting DITA map file is completely DITA-compliant, although within the FrameMaker authoring environment some additional elements are added to provide a more efficient authoring experience. These elements have an “fm-” prefix.

DITA maps are the fundamental mechanism provided by DITA for organizing topics into deliverables (PDFs, online Help, HTML, etc.). A separate map can be created for each deliverable type or a single map can be used for multiple deliverables. Maps organize the topics into a logical hierarchy using topic referencing elements (<topicref> for a “map” and <chapter>, <appendix>, <topicref>, and others for a “bookmap”). A topic referencing element can also reference other maps, allowing you to create multiple levels of nested maps as appropriate for your topic organization and workflow.

All DITA map files (both map and bookmap) must use the “.ditamap” file extension. This is a requirement enforced by the DITA Open Toolkit and is also required by DITA-FMx.

On the opening of a DITA map file, some topic referencing elements may be updated to include a child <fm-reflabel> element that displays a label within a locked text range. Other topic referencing elements may already contain a <navtitle> element which serves this same purpose. When opening a map that has no <navtitle> elements, the settings in the Map Options dialog determine if <navtitle> elements are added or not.

The Auto-load Topicrefs option (in DITA Options) controls what is displayed for the <fm-reflabel> and <navtitle> elements (titles, file names, or both).

RELATED INFORMATION:

"Generate Book from Map" on page 75
"Book Build Settings" on page 57
"Setting up Book Builds (PDF)" on page 59

Basic Map Structure

Basic information on working with DITA map files.

The basic DITA map file uses the <map> element as the root node. The <map> element can be followed by a <title> element. After the title is an optional

<topicmeta> element which contains various elements that define metadata for the map. This metadata may define copyright or other legal information regarding the publication, as well as keywords or online Help IDs. The type of metadata used in a map will generally be defined by the deliverable format and the method the output is generated.

The only topic referencing element in a <map> is <topicref>. A map can include the <keydef> element which is used to create key definitions (for use in key-based referencing).

Insert a <topicref> element and associate it with the target topic. To create a topic hierarchy, insert new <topicref> elements as children of a parent <topicref>. If you are creating a FrameMaker book from a map, keep in mind that the "top-level" topicrefs (those that are immediate children of the root element) will become chapter FM files. Any child topicrefs will be added to the chapter files as sub-topics.

One or more relationship tables can be added after the topicrefs. A relationship table defines relationships between topics, and is the preferred method for defining topic linking (over the use of inline cross-references). Relationship tables are defined by the <reltable> element (which is represented by a standard FrameMaker table), with <topicref> elements inside the table cells. You can create relationship tables with any number of columns, but typically they will use two or three columns.

The simplest and easiest to understand is the 2-column unidirectional reltable, where the topics (defined by topicrefs) in the left column will link to the topics in the right column. To create this type of relationship table, insert a <reltable> with two columns (the number of rows doesn't matter, you can add more as needed). Then locate the first <relcolspec> element (this will control the first column's specifications), and set the linking attribute to "sourceonly." Locate the second <relcolspec> element and set the linking attribute to "targetonly." The selected values will display in the table heading (along with the value of the relcolspec/@type attribute if provided). Insert <topicref> elements in the table cells to create related-links between the specified topics. If you create a relationship table without setting the relcolspec/@linking attributes, the resulting links will be bidirectional.

Bookmaps

Information specific to bookmaps in DITA-FMx.

The <bookmap> element allows you to create a hierarchy of topic references that resembles the structure of a printed book. The general order of elements in a <bookmap> is similar to that of the <map>, although the top-level element names will be different. The title of a <bookmap> can be <title> or <booktitle>.

The <bookmap>'s metadata is stored in the <bookmeta> element (very similar to the <topicmeta> of the <map>).

The most significant difference between the <map> and <bookmap> are the topic referencing elements. A <bookmap> provides many specialized topic referencing elements for book-specific purposes that group topicrefs into logical sections.

The first logical grouping is the <frontmatter> element. The <frontmatter> element can contain a number of topic referencing elements (including <topciref>) that specify topics that are part of a book's frontmatter. A similar <backmatter> element can be added at the end of the book. One of the special elements in the <frontmatter> and <backmatter> is the <booklists> element.

The <booklists> element (a child of the <frontmatter> and <backmatter> elements) can contain one or more "list" elements that are intended to provide generated lists (similar to the FrameMaker generated list files like a "toc" or "index"). Using DITA-FMx, when you insert an element that is a child of <booklists>, you are not prompted for a target file name. Instead this element is left empty, and doesn't reference any file at all. At book-build time, if you have enabled the "Replace List Files with Generated Lists", those elements will become FrameMaker generated lists (assuming you've set up the proper "component templates").

Following the <frontmatter> element can be a number of topic grouping elements such as <part>, <chapter>, and <appendix>. The <part> element can be used to organize <chapter> and <appendix> elements into parts, and the <chapter> and <appendix> elements are used to organize <topciref> elements. After the last <part>, <chapter>, or <appendix> can be the <backmatter> element, similar to the <frontmatter> element described above.

A <bookmap> can also make use of relationship tables in the same way they are used in a <map>. Note that even though your <bookmap> may make use of <part>, <chapter>, and <appendix> topic referencing elements, the <reltable> can only contain <topciref> elements.

Best Practice for Book Assembly

Tips and information that applies to both maps and bookmaps.

When setting up a <map> or <bookmap> that will be used to define a book-like deliverable (either as a PDF or online format that breaks topics into sections or chapters), it is common to create a root map with submaps (or chapter-maps). The root map would, at a minimum, contain <topicref> (or <chapter>, <appendix>, etc.) elements that point to each submap (chapter). If your root map is a <bookmap>, it would also contain the <frontmatter> and <back-matter> along with the appropriate child elements. It might also contain a relationship table, but depending on your structure, you might want to maintain relationship tables in the submaps.

The submaps should be standard DITA maps (not a bookmap). If you are using the default DITA-FMx XSLT import script that is provided with the Book application, your chapter maps should contain a single root <topicref> element which defines the chapter title and any optional content that would appear before the first H1-level heading in the generated FM file. Any topicref elements that are children of the root topicref become the H1, H2, and so on, headings within the chapter.

NOTE: *When using the default DITA-FMx XSLT import script, the use of a submap adds no hierarchy or topics to the generated FM files. A map should be thought of as purely an authoring convenience. All headings in the resulting FM files are created from titles in DITA topics. Currently, the <topichead> and <topicgroup> elements are ignored in the book-build process, as are any attributes applied to those elements.*

If you want to be able to use the titles from the submaps as the chapter titles, you'll need to modify the XSLT import script to pull that content from the map and insert it into the proper location in the output.

This method of creating chapter maps makes it very convenient to assign a "chapter" to a specific writer. Note that it is perfectly reasonable to take this nested map concept to further levels. You may have components within a chapter that make sense to group into a map. This is especially useful if you reuse these components in other maps or deliverables.

Recommended Folder/File Structure

Tips for setting up folders and files that will provide the desired results when generating books and other types of output.

While, in theory, you should be able to use any folder structure for your maps and topic files, the default DITA-FMx Book application's XSLT import script is set up to work best with a specific structure as described below.

- First and foremost, all files must be on the same "drive," and should be referenced with relative path names. If you're working on a shared server, be sure to always access the files via a mounted drive letter. If you ever see drive letters or UNC path descriptors ("\\SERVERNAME\PATH\..."), stop and fix things so that you're only seeing relative paths in the source files.
- Your root map should be in a parent folder or the same folder as the topic or map files it references; a map should not reference a topic through a folder "above" itself (the @href attribute should never start with "../"). Although this will often work, you may run into situations that it won't,

and if you're planning on generating CHM or other compiled Help output through the DITA-OT, the map must always be at the root of the project.

- Any submaps should also be in the same top-level folder as the root map. This isn't a hard requirement, and will work in most cases with submaps in other folders, but if the folder structure is very complex, the reference resolving process may fail.
- Images and conrefs referenced by topics in the project may be in folders above the root map, however if is best to keep these in folders that are siblings or children of the root map's directory.

The recommended folder structure for multiple projects is as follows:

- A top-level folder (*dita-projects*)
- Within the top-level folder are folders for the shared content and the project folders themselves. Such as, *shared-images* (for images that are shared among the projects), *shared-content* (for conref source that is shared among the projects). If you have shared topics, those may work in this folder as well, but certain output types may not work well with that structure.
- Within each project folder is a folder for topics (just one) and a folder for images, as well as one or more "book" output folders (one per book type).
- Also in the project folder is the root map (possibly with an underscore prefix so it sorts to the top) and all submaps. This puts all of the maps in a separate folder from the topics and makes them easily accessible.
- Within each book output folder is that output type's *component-templates* folder and the *ditafmx-bookbuild.ini* file. This ensures that building a book to that folder will always result in the same output. If all of your books use the same component templates, you may want to have a shared component templates folder.

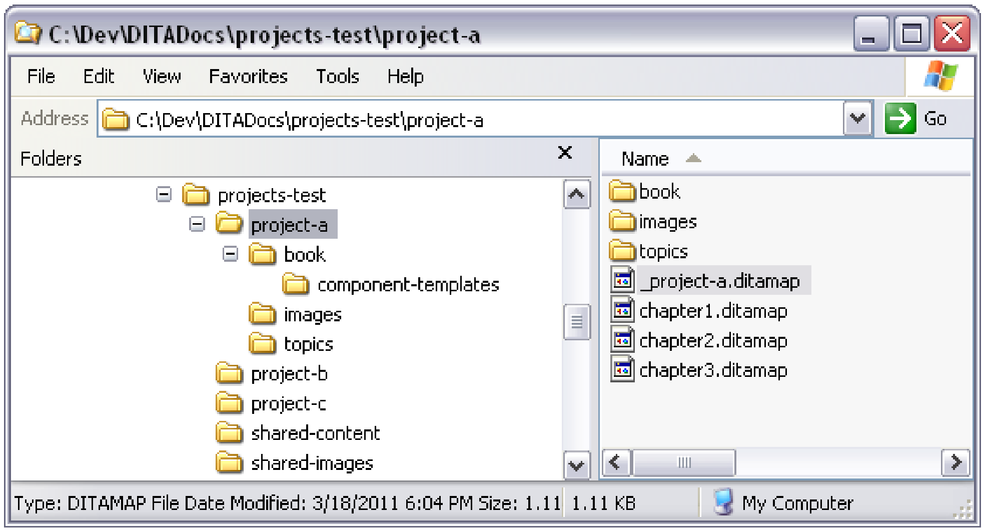


Figure 1-12: Recommended folder structure

Using Keyspaces in DITA-FMx

In DITA-FMx, a keyspace consists of all keys defined in a root map and all submaps, optionally filtered by an associated ditaval file.

The implementation of keyspaces in DITA-FMx is a bit different than that in default FM-DITA (and perhaps other tools as well).

Each registered key space is stored as a ".keyspace" file in the user's DITA-FMx folder (**DITA-FMx > Open DITA-FMx Folder**). The format of this file is XML, and contains all of the necessary data (normalized and filtered) to insert and display key references for that key space. This is not a DITA file, so don't try to open it using a FrameMaker DITA structured application. If you're curious, take a look by opening it in a text editor; the structure will be apparent.

When you create a map and start adding key definitions to it, use the Keyspace Manager to register the root map. Consider making use of the Keyspace Generation feature (see Keyspace Options) so that the keyspace file is kept in sync with the key definitions in the map.

If you are using submaps, remember to register the root map, not the submap; the key space should always be defined by the root map. As a convenience feature, DITA-FMx will automatically update the keyspace, even when working on a submap, if the submap includes an "fmx-root-map" <othermeta> element. This feature is described in Keyspace Options.

If you are working with multiple projects and thus multiple keyspaces, consider using the Keyspace Resolution feature (also described in Keyspace Options) to automatically change the default keyspace when the root map or an associated submap is opened.

When registering a keyspace, you have the option to assign an associated ditaval file. If you are setting up key definitions that are modified by ditaval filtering, you may want to select one of the filtering scenarios as the default for editing. You can always change the ditaval file associated with a keyspace to switch filtering schemes if needed. If no ditaval is assigned to a keyspace, all key definitions are included in that keyspace (duplicate keys are ignored; first key wins).

Only registered ditaval files are available to be assigned to a keyspace. Use the Ditaval Manager to register a ditaval file.

When publishing through the DITA-FMx Generate Book from Map command, the ditaval file selected in the Book-Build Options dialog or specified in the book-build INI file will be applied to the keyspace regardless of the current default keyspace configuration.

IMPORTANT: *In order to use keys, you must be using DITA 1.2 or greater. This is determined by the ditaarch:DITAArchVersion attribute on the "topic" element. The DITA version is not typically assigned explicitly, but has a default value that will affect the availability of these features in DITA-FMx.*

RELATED INFORMATION:

"Keyspace Options" on page 51
"Keyspace Manager" on page 20
"Insert Key Element Reference" on page 27

Working with Keys

Keys can provide a huge benefit but also create great complications. Here are some tips that may help to reduce some of the latter.

- When setting up a project, remember to use the Keyspace Manager to register a keyspace for the root map. Do not register keyspaces for the submaps (unless those are used as a "root map" for other deliverables); only the root map's keyspace is used for resolving keys.
- If your key definitions are not in the root map, but in one or more submaps, be sure to use the special DITA-FMx "fmx-root-map" other-meta flag to reference the root map from the submaps. This feature (along with use of the Keyspace Generation feature) ensures that any edits to the key definitions will be automatically included in the keyspace.

- When creating a map, go ahead and assign a key to each of the significant topicrefs. They may not get used, but may provide an easy way to reference those topics. This is especially useful for references if each command or function has its own key.
- It doesn't matter where keys are defined in a map. Some conventions will have them at the "top" of a map, but this is not necessary. The only thing to keep in mind is that if you have multiple instances of the same key, the first one (in document order) wins.
- Rather than cluttering up your root map or submaps used to define topicref hierarchy, consider adding <keydef> elements to a "keymap." This is not a map element type, but rather a map that you use to store key definitions. You can then reference this keymap from your root map using a <mapref> element. Add the <mapref> element to your <frontmatter> or other top-level container.
- A "key element reference" is defined by element content contained within the <topicmeta> in a <keydef> in a map (or by a keyref to a glossary entry). Insert a <keydef> and provide the key name; the referenced file is not required if you are creating a key element reference. After inserting the <keydef> add a <topicmeta> element then a nested <keywords> element and a <keyword> element. The content of the key element reference will be taken from the <keyword> element within the <keydef>'s topicmeta.

  Assign filtering attributes to the <keyword> elements to change the value of the resulting key element reference based on the use of different ditaval files.

NOTE: The term "key element reference" is specific to DITA-FMx. We could find no common term for this type of a key-based reference, so use this term.

RELATED INFORMATION:

"Using Keyspaces in DITA-FMx" on page 51
"Keyspace Options" on page 51
"Keyspace Manager" on page 20

Working with Glossaries

Providing a glossary can significantly enhance most any documentation set.

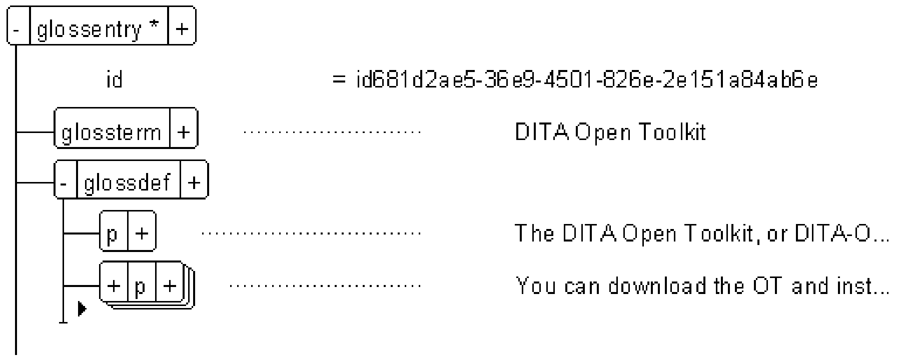
RELATED INFORMATION:

"Glossary Entries" on page 54
"Glossary Lists" on page 56

Glossary Entries

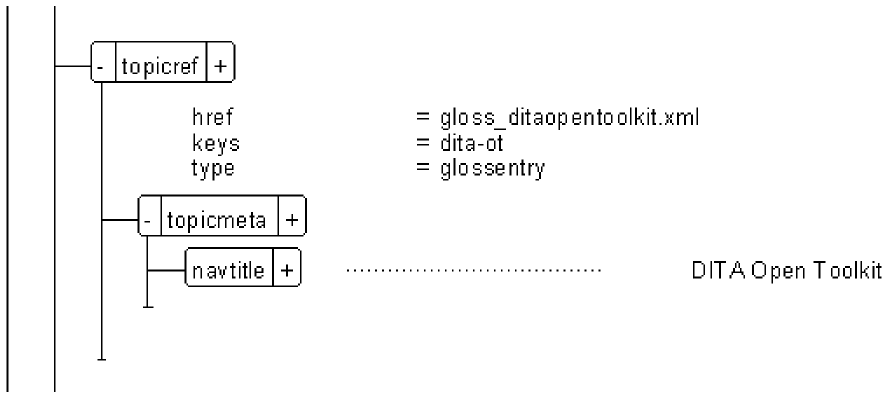
Glossary entries are an essential part of a glossary.

In addition to supporting the basic glossary models, DITA-FMx supports various aspects of key-based referencing of glossary terms. The following image shows a simple <glossentry> topic that defines only <glossterm> and <glossdef> elements.



```
id = id681d2ae5-36e9-4501-826e-2e151a84ab6e
DITA Open Toolkit
The DITA Open Toolkit, or DITA-O...
You can download the OT and inst...
```

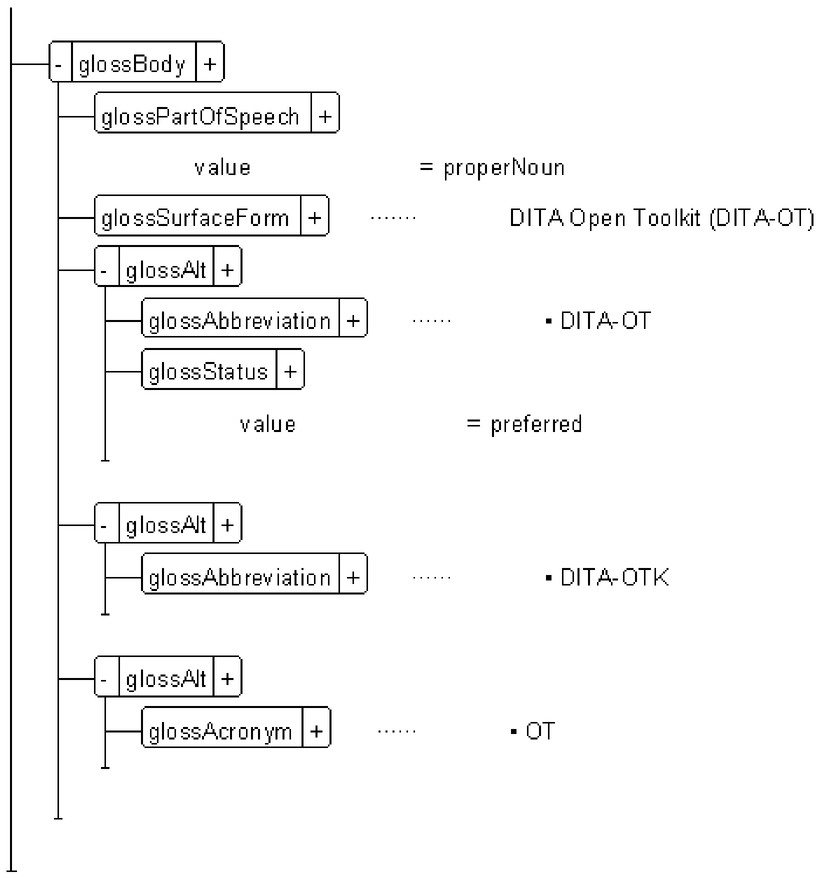
When including a <glossentry> in a map, you might reference it with a <topicref>. If the @type attribute is set to “glossentry”, and the @keys attribute is set, you can insert references to the glossary term through the **Insert Key Element Reference** command.



```
href = gloss_ditaopentoolkit.xml
keys = dita-ot
type = glossentry
DITA Open Toolkit
```

Including a <glossBody> in your <glossentry> topic lets you provide additional information about the term. If the term can be represented by “complete” and abbreviated labels, you might consider using the <glossSurfaceForm> element to define the “full form” (which includes the primary abbreviation in parenthesis). Then one or more <glossAlt> groups to define the primary and alternate shorter versions.

The <glossAlt> element contains one of <glossAbbreviation>, <glossAcronym>, <glossShortForm>, or <glossSynonym>, optionally followed by a <glossStatus> whose @value attribute indicates that the term is preferred, restricted, prohibited, or obsolete.


- glossBody +
glossPartOfSpeech +
value = properNoun
glossSurfaceForm + ······ DITA Open Toolkit (DITA-OT)
- glossAlt +
glossAbbreviation + ······ ▪ DITA-OT
glossStatus +
value = preferred
- glossAlt +
glossAbbreviation + ······ ▪ DITA-OTK
- glossAlt +
glossAcronym + ······ ▪ OT

If the “Glossary Term Swapping” book-build option is enabled, the first instance of a reference to a term in a chapter (generated FM file) will be the content from the <glossSurfaceForm> element, and later instances will be the first or “preferred” <glossAlt> entry (or the <glossterm> if no <glossAlt> elements are defined).

NOTE: *The “Conditionalize Data ...” option in the DITA Options dialog, applies the DITA-Data condition to all <data>-based elements, including <glossPartOf-Speech>, <glossStatus>, <glossProperty>, and others.*

RELATED INFORMATION:

www.oasis-open.org/committees/down-load.php/35292/DITA%201.2%20KeyRef%20Feature%20Description_-Final.pdfDITA_1.2 Keyref: Feature Description - 21 September 2009 (Sowmya Kannan)DITA_1.2 Keyref: Feature Description - 21 September 2009 (Sowmya Kannan)DITA...

DITA 1.2 Keyref: Feature Description - 21 September 2009 (Sowmya Kannan)

www.oasis-open.org/committees/download.php/34831/GlossarySpecialization-BestPractice_Final.pdfDITA_1.2 Glossary and Terminology Specialization Feature Description_- 21 October 2009 (Kara Warburton)DITA 1.2 Glossary and Terminology_Specialization Featu...

DITA 1.2 Glossary and Terminology Specialization Feature Description - 21 October 2009 (Kara Warburton)

www.oasis-open.org/committees/download.php/35766/glossary_and_term_management.pdfDITA_1.2 Feature Description: Improved glossary and terminology handling_- 16 December 2009 (Tony Self)DITA 1.2 Feature Description: Improved_glossary and terminology ha...

DITA 1.2 Feature Description: Improved glossary and terminology handling - 16 December 2009 (Tony Self)

www.oasis-open.org/committees/download.php/38692/DITA1.2FeatureDescriptionAcronym.pdfDITA_1.2 Feature Article: Acronym Best Practices - 24 May 2010 (DITA_Translation Subcommittee)DITA 1.2 Feature Article: Acronym Best_Practices - 24 May 2010 (DITA Tr...

DITA 1.2 Feature Article: Acronym Best Practices - 24 May 2010 (DITA Translation Subcommittee)

"Using Keyspaces in DITA-FMx" on page 51
"Working with Keys" on page 52
"Book Build Settings" on page 57

Glossary Lists

Once you've developed the glossary entries, you may want to present them as a generated list for easy browsing.

The assumed use for this list is as an additional "TOC" list at the beginning of the book. This would be presented as a list of terms that link to the full glossary topics (glossentries). With DITA-FMx, we can leverage FrameMaker's ability to create multiple types of generated lists, and a glossary list is just one of those types.

The following steps describe one way to achieve this list.

1) We assume that you're using a bookmap, and in the <frontmatter> <booklists> element you should insert an empty<glossarylist> element.

2) Next, set up a "gentpl" component template to properly format that generated list (like a TOC or Index). You might start by cloning the *gentpl~toc.fm* file to *gentpl~glossarylist.fm.*

3) In your bookbuild INI, you need to provide the para tag names for "GeneratedFile-glossarylist" like you do for the TOC.

4) Not required, but a good practice would be to separate your <glossentry> topics into a separate submap that has a single root element so they all show up as child nodes of the root.

```
<map>
<title>Glossary</title>
<topicref href="glossmain.xml">
```

```
<glossref href="glossentryone.xml"/>
<glossref href="glossentrytwo.xml"/>
<glossref href="glossentrythree.xml"/>
...
</topicref>
</map>
```

5) In your template, make sure the <glossterm> (titles) get tagged with some unique para tag, so you can use that in step #3 (above).

RELATED INFORMATION:

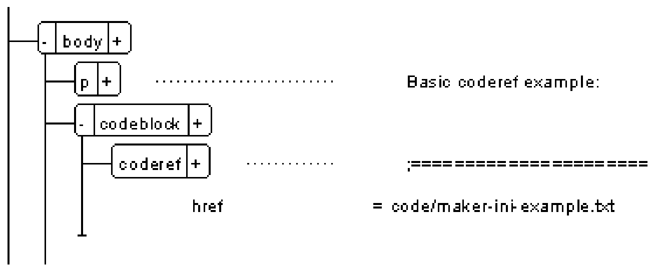
"Glossary Entries" on page 54
"The Book Build Settings Dialog and the Book-Build INI File" on page 61
"Working with Maps" on page 45

Using coderefs

The coderef element lets you directly reference a text file in your documentation.

A <coderef> is basically a conref to a non-DITA file. If your documentation contains code samples, you can use a <coderef> to include code samples from the actual code files rather than copying and pasting that content into your document. This allows you to include real, working (and tested) code samples in your documentation rather than running the risk of accidentally modifying that code making it invalid.

To use a <coderef>, insert the element into a <codeblock> element. It can be the only element in the <codeblock> or you can add other content including other <coderef> elements.



The content of the referenced code file is inserted into FrameMaker as a locked range (like a text inset).

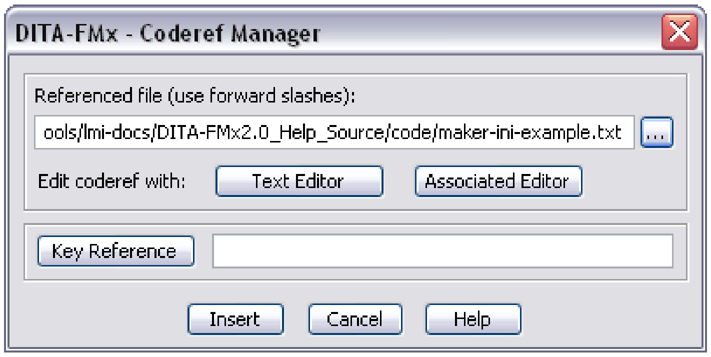
Basic coderef example:

```
;=============================
; FMx 1.1
;ditafmx=Standard, ditafmx, DITA-FMx\ditafmx_110.dll, structured
;ditafmx_app=Standard, ditafmx_app, DITA-FMx\ditafmx_110_app.dll, structured

; FMx 2.0
ditafmx=Standard, ditafmx, DITA-FMx-2\ditafmx_110.dll, structured
ditafmx_app=Standard, ditafmx_app, DITA-FMx-2\ditafmx_110_app.dll, structured
;=============================
```

Any tabs within the code file are converted to spaces. By default each tab converts to 4 spaces, but you can set the value in the Authoring Options dialog.

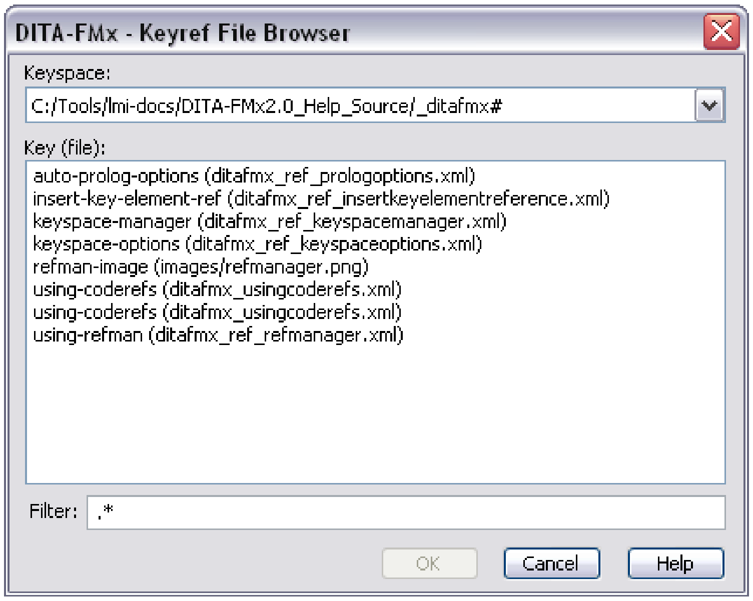
Double-click the <coderef> to access the Coderef Manager dialog.



Referenced file (use forward slashes): ools/lmi-docs/DITA-FMx2.0_Help_Source/code/maker-ini-example.txt

Figure 1-13: DITA-FMx Coderef Manager dialog

Through this dialog, you can edit the code file or modify the referenced file. Use the Key Reference button to select a key to use for referencing the code file.



Keyspace: C:/Tools/lmi-docs/DITA-FMx2.0_Help_Source/_ditafmx#

Key (file):

```
auto-prolog-options (ditafmx_ref_prologoptions.xml)
insert-key-element-ref (ditafmx_ref_insertkeyelementreference.xml)
keyspace-manager (ditafmx_ref_keyspacemanager.xml)
keyspace-options (ditafmx_ref_keyspaceoptions.xml)
refman-image (images/refmanager.png)
using-coderefs (ditafmx_usingcoderefs.xml)
using-coderefs (ditafmx_usingcoderefs.xml)
using-refman (ditafmx_ref_refmanager.xml)
```

Filter: .*

Figure 1-14: DITA-FMx Keyref File Browser dialog

By default, when selecting a key reference for a <coderef>, the Keyref File Browser displays keys that reference all file types. If you want to browse for specific file types, edit the Filter field in the dialog. This field can be set to a space delimited list of one or more file extensions (no wild cards), such as ".txt .java .js".

RELATED INFORMATION:

"Authoring Options" on page 47

Setting up Book Builds (PDF)

Issues and options regarding the generation of a FrameMaker book file from a DITA map or bookmap.

DITA-FMx provides numerous options for controlling the features and properties of a book and its components when using the Generate Book from Map command. There are three areas that affect the way a book is created, the Book structure application, the Book Build Settings, and the *book-build INI file*.

RELATED INFORMATION:

"Book Build Settings" on page 57
"Book-Build INI file" on page 62
"Generate Book from Map" on page 75
"Developing Custom Structure Applications" on page 21
"Customizing the Formatting of the Default Structure Applications" on page 24
"Working with Maps" on page 45
"Merge Para Tags" on page 13

The Book Structure Application

The Book structure application defines the fundamental structure and formatting that is applied to the topic files that become FM chapter files through the book-build process.

Because it's possible to combine multiple topic types into a single chapter file, this structure application must contain the element definitions for all possible elements and structures that are included in the final book.

If you are using "ditabase" (the DTD that supports all of the basic topic types in a single DTD), it will be virtually identical to the Topic structure application. However, if you are using the "Doctype/Application Mapping" option which lets you use individual DTDs (and structure applications) for each topic type, you will need to create a special Book application that supports all of those topic types in a single EDD. The default Book application files are installed to *FrameMaker\Structure\xml\DITA-FMx_1.2\Book*.

The Book structure application definition (**StructureTools > Edit Application Definitions**) also specifies an “Import” XSLT file which is the first step of the conversion process. This XSLT is the piece that does the aggregation of DITA XML files into chapter-based FM files. It processes the root map, any sub maps, and all topic files to build a single XML file that is opened in FrameMaker and becomes the book and components.

If you look through this file (*bookmap2fmbook.xsl*), you’ll see that it adds FM-specific processing instructions that tell FrameMaker where the book file starts and where each of the chapter FM files start. You can modify this XSLT to accomplish additional custom processing as needed. One XSLT modification that may prove easy and useful is to pass metadata from the map or bookmap so it is added to the <fm-ditabook> element as attributes. These attribute values can then be used in various ways in the resulting generated book and chapter files. For more information on this see, Passing Map-level Metadata to the FM Book.

The resulting generated FM book and components are set up so that each XML file is represented by an <fm-ditafile> element. This element is used to provide a container on which to hang file-specific attributes, specifically the href attribute which is the path and filename to the original DITA topic file. This is needed for later processes to properly resolve references. Another attribute that is found on the <fm-ditafile> element is the mapelemtype attribute. The mapelemtype attribute is set on the top-level <fm-ditafile> elements and will have the value of the element name of the associated topic referencing element in the map. For example, a <chapter> element will result in an <fm-ditafile> element with a mapelemtype attribute set to ‘chapter’.

The Book Build Settings Dialog and the Book-Build INI File

Describes the methods available for generating a FM book from a DITA map.

The Book Build Settings dialog (**DITA-FMx > Options**) provides a number of options that control the processes that are run on the generated book file and components after the initial aggregation has taken place. If all of the options are deselected, you’ll end up with an unprocessed book file and FM files. This is typically not what’s needed (since any references will not be resolved, among other things), but if you have special processing needs, this may be a viable option.

The *book-build INI file* (*ditafmx-bookbuild.ini*) is a user-managed text file that defines the properties (pagination, numbering, and formatting) of each book component including the properties of generated list components (toc, index, etc.). This file also defines additional settings that control how book-level variables are used, as well as any pre and/or post build scripts. Settings defined in the Book Build Settings dialog may be overridden in the book-build INI file. This file is intended to allow the user to predefine all of the properties for each book-build, to ensure that subsequent builds result in the exact same output.

You would typically have one book-build INI file for each book that is created. The **Generate Book from Map** command looks for the book-build INI file in the folder selected for the book file to be created. If one does not exist in that location, it reads the settings defined in the user's DITA-FMx folder (**DITA-FMx > Open DITA-FMx Folder**).

The first option in the Book Build Settings dialog is "Normalize Reference Paths." This option processes all elements that contains an href or conref attribute and converts them into absolute paths based on the new relationship of content due to the file aggregation. If this option is selected, the "Add Related Links" and "Reload References" are also available. If "Add Related Links" is selected, any related links defined in reltable elements will be built and added to the bottom of each topic as appropriate. The "Reload References" option reloads and updates the references based on the updated reference locations.

Other options are described in detail in the Book Build Settings topic. Three of these options rely on properties that can be specified in a *ditafmx-bookbuild.ini* file. This INI file must be created manually and placed in either, the folder that contains the generated book file, or the user's DITA-FMx folder (**DITA-FMx > Open DITA-FMx Folder**). This file is initially checked for in the book folder, if it does not exist there one will be used from the user's DITA-FMx folder. This lets you maintain properties and templates that are specific to each book. The details of this INI file are provided in the Book-Build INI file topic.

The "Assign Numbering and Pagination" option enables the "NumberingFirst" and "NumberingDefault" sections of the *ditafmx-bookbuild.ini* file. This INI file can contain either or both of these "Numbering" sections for each mapelemtype attribute (each unique topic referencing element in the map file). Within each section are options that mirror the standard FrameMaker book numbering and pagination options. A simple bookmap that contains a "toc", a few chapters, and an index (indexlist) would need the following four "Numbering" sections:

- NumberingFirst-toc
- NumberingFirst-chapter
- NumberingDefault-chapter
- NumberingFirst-indexlist

If only one element of any given map element type exists in the bookmap, you only need the NumberingFirst section for that element type, otherwise you should include both the NumberingFirst and NumberingDefault sections. These numbering options can be used for both DITA maps and bookmaps.

If enabled, the “Replace List Files with Generated Files” option will replace the placeholder files in the frontmatter and backmatter elements with FrameMaker generated lists. This uses the “GeneratedFile” sections in the *ditafmx-book-build.ini* file. You should create a GeneratedFile section for each of the unique list files. These sections specify the component type and the tags that are used for the generated list. Additionally the General section specifies the BookTemplatesDir option which specifies the folder that contains the template to use for each of the generated lists. The BookTemplatesDir location can be an absolute or relative path, if relative, it is relative to the book folder. For example, a book with a toc and index would specify two GeneratedFile sections:

- GeneratedFile-toc
- GeneratedFile-indexlist

The INI code would look something like the following:

```
[GeneratedFile-toc]
ComponentType=Toc
NumTags=3
1=title.0
2=title.1
3=title-index

[GeneratedFile-indexlist]
ComponentType=IndexStandard
NumTags=1
1=Index
```

The valid ComponentType values are specified in the Book-Build INI file topic. The generated list templates in the BookTemplatesDir folder must be named with the file naming convention of *gentpl~<mapelemtype>.fm* (that’s the string “gentpl” followed by a tilde, then the associated map element type name, with a “.fm” file extension). The “Replace List Files...” option is only valid with DITA bookmaps. (Sample templates are available in the *DITA-FMx_1.2\Book\component-templates* folder.)

The default XSLT import script makes the book title (from map/title, map/@title, bookmap/title, or bookmap/booktitle/mainbooktitle) available as a header/footer variable in files generated from DITA topic files. If you want to include the book title in generated list files, create a variable named “FMxBookTitle” and insert it into the appropriate location in your generated list template file. The variable will be updated when the generated lists are added to the book. If you’d like to use a different variable name, you can specify that name in the BookTitleVariableName parameter in the INIOnly section of the *ditafmx.ini* file.

The “Apply templates” option applies component templates to the other (non-list) files. It also uses the folder specified by the BookTemplatesDir option. The file naming convention for component templates is *tpl~<mapelemtype>.fm*. This option is also only applicable for DITA bookmaps. (Sample templates are available in the *DITA-FMx_1.2\Book\component-templates* folder.)

If you'd like to perform additional automated processing to the generated book and component files, you can use the "Run Custom Script" option to specify one or more FrameScript or FDK clients to run near the end of the book-build processing. These scripts run before pagination has been completed (conditional text has not been hidden and other processes that affect pagination are yet to be run). If you want to run a script after final pagination has been applied, use the RunPostPaginationScript settings in the *ditafmx-bookbuild.ini* file.

Variable values and condition states can be defined by values in the DITA map. These are controlled by attributes on the <fm-ditabook> element. For more information, see Adding Map to Book Metadata Mappings.

There are a number of other book-build settings that are controlled by the *ditafmx-bookbuild.ini*; please review the Book-Build INI file topic for more details.

A sample *ditafmx-bookbuild.ini* is provided in the *DITA-FMx* installation folder. This file is also provided in the *DITA-FMx_Help_Source.zip* file along with the working templates.

Using the outputclass attribute as a "mapelemtype" value

You can set the outputclass attribute on topicrefs or topicref-based elements in a map and that value will be assigned to the outputclass attribute on the associated fm-ditafile element. This mapping is used when the UseOutputclassForType parameter is set to 1 in the book-build INI file. When this feature is enabled, this value will be used instead of the fm-ditafile/@mapelemtype attribute that is set based on the type of the associated map element. For additional information see "UseOutputclassForType" in the Book-Build INI file topic.

Passing Map-level Metadata to the FM Book

Use attribute and element values in the map to affect formatting and properties in the structure application, set conditional hide/show settings, and update variable definitions.

DITA-FMx provides a number of features which make it easier to use map-level metadata to control formatting and data in the generated chapter files. Attribute and element values in the map are passed to the <fm-ditabook> element (the root of the generated book file) during the XSLT import step of the book-build process. Once these attribute values are set on the <fm-ditabook> element they can control other aspects of the component document formatting and content.

These attributes can be used in various ways in a structure application EDD and template and to set the hide/show state of conditional text. They can also control the value of variable definitions in files that are included in the generated book file.

By default, only a limited set of metadata (attributes and element values) are transferred to the generated book. You can add additional metadata by modifying the Book application EDD, the *fmx-book_1.2.dtd* file, and the Book application import XSLT file (*bookmap2fmbook.xsl*), as described in Adding Map to Book Metadata Mappings. The following tables describe the metadata supported in the default files.

Bookmap metadata (in bookmeta)	**If multiple, use ..**	**fm-ditabook attribute**
publisherinformation/organization	first instance	@pubinfo-org
publisherinformation/published/completed/year	first instance	@pubinfo-completedyear
publisherinformation/published/completed/month	first instance	@pubinfo-completedmonth
publisherinformation/published/completed/day	first instance	@pubinfo-completedday
permissions/@view	first instance of permissions	@permissions-view
category	first instance	@category
critdates/created/@date	first instance	@created
critdates/revised/@modified	last instance	@revised
bookid/bookpartno	first instance	@bookpartno
bookid/edition	first instance	@edition
bookid/isbn	first instance	@isbn
bookid/booknumber	first instance	@booknumber
bookid/volume	first instance	@volume
bookid/maintainer/person	first instance	@maintainer-person
bookid/maintainer/organization	first instance	@maintainer-org
bookrights/copyrfirst/year	-	@copyrfirst

Bookmap metadata (in bookmeta)	If multiple, use ..	fm-ditabook attribute
bookrights/copyrlast/year	-	@copyrlast
bookrights/bookowner/organization	first instance	@copyrholder
bookrights/bookowner/organization	first instance	@bookowner-org
bookrights/bookowner/person	first instance	@bookowner-person
bookrights/bookrestriction/@value	-	@bookrestriction
prodinfo/prodname	-	@prodname
prodinfo/vrmlist/vrm/@version	last instance	@version
prodinfo/vrmlist/vrm/@release	last instance	@release
prodinfo/vrmlist/vrm/@modification	last instance	@modification
authorinformation/personinfo/namedetails/ personname/firstname and lastname	first instance	@authorname

Map metadata (in topicmeta)	If multiple, use ..	fm-ditabook attribute
critdates/created/@date	first instance	@created
critdates/revised/@modified	last instance	@revised
copyright/copyryear/year	first instance	@copyrfirst
copyright/copyryear/year	last instance	@copyrlast
copyright/copyrholder	-	@copyrholder
author	-	@authorname

Structure application template and EDD context rules

Attribute value building blocks can reference an attribute in the book file (as well as attributes in elements in the same file). Use the following building block to reference the value of the prodname attribute on the <fm-ditabook> element:

```
<$attribute[prodname:fm-ditabook]>
```

Attribute value building blocks like this can be used in cross-reference formats, header/footer variables, as well as EDD prefix and suffix rules. You can also use this type of building block in EDD context rules to apply formatting based on the values of book-level attributes. A context rule could start as follows:

```
If context is: * < fm-ditabook[prodname="DITA-FMx"]
```

Note that changing an attribute value directly in the book doesn't affect the component files until that book is updated.

Control conditional text hide/show state

To enable the setting of conditional text hide/show states using book-level attributes you must be using a book-build INI file for your book building process. If you're not familiar with the book-build INI file, you should review the information in The Book Build Settings Dialog and the Book-Build INI File. To enable this feature, set the ImportAttrsAsConds parameter to "1" in the General section of the book-build INI file.

When using this feature, conditional text names must follow a specific naming convention of "<*prefix*><*attribute*>-<*value*>". Where <*prefix*> is "fmx-" by default, but can be changed by setting the General/AttrAsCondPrefix parameter, <*attribute*> is the name of the fm-ditabook attribute, and <*value*> is the value of that attribute. For example, if you wanted to be able to hide/show a condition that was controlled by the value of the bookmeta/bookrights/bookrestriction/@value attribute, you could use conditions named "fmx-bookrestriction-confidential" and "fmx-bookrestriction-restricted". These conditions must exist in the templates (Book application template and any component templates) and must be set to "Show" (if they are set to "Hide", any content tagged with that condition will be lost when the template is applied).

To control the hide/show state of a condition, you must add the condition names to the ConditionMap section of the book-build INI file. List each condition name followed by "Hide" or "Show" (typically "Show").

```
[ConditionMap]
fmx-bookrestriction-confidential=Show
fmx-bookrestriction-restricted=Show
```

When this feature is enabled, DITA-FMx starts off by hiding all "fmx-" conditions (or whatever is defined as your prefix using the AttrAsCondPrefix parameter) in the document. Then, each fm-ditabook attribute and value are matched with the condition named in the ConditionMap section. If a match is found (using the "<*prefix*><*attribute*>-<*value*>" naming convention), that condition is set to the specified hide/show state. The <fm-ditabook> attribute value set in the bookmap must be represented by a condition specified in the book-build INI, so be sure to include all possible condition names that you're hiding and showing and set them all to "Show" (even if some may need to be hidden). Only one value for an attribute can be set at a time, so only the condition representing an attribute value match will show.

If you need to work in reverse and have all of the conditions set to "Show" by default, you can set the General/AttrAsCondDefaultState parameter to "Show", then set the values in the condition map to "Hide".

This condition setting works in both generated (structured) FM files as well as unstructured FM files (from generated lists as well as those files that have been "included" via the book-build INI).

Update variable definitions

To enable the updating of variable definitions using book-level attributes you must be using a book-build INI file for your book building process. If you're not familiar with the book-build INI file, you should review the information in The Book Build Settings Dialog and the Book-Build INI File. To enable this feature, set the ImportAttrsAsVars parameter to "1" in the General section of the book-build INI file.

If the ImportAttrsAsVars parameter is set to "1", all <fm-ditabook> attributes are used to update like-named variables in each book component (both structured and unstructured files). Like the conditional text names, the variable names must adhere to the naming convention of "fmx-*ATTRIBUTE-NAME*". The value of the variable is set to the value of the <fm-ditabook> attribute.

In order to make use of these map-defined variables, create a variable definition of the proper name (following the naming convention described above), give it some default value, then insert it into the document. If you're using this in a structured application template file, you'll be placing this variable in a frame that's not part of the main flow (since the flow is replaced with the XML-defined data). If this is being used in an unstructured file that is being included (using the IncludeFiles book-build INI feature), you can place the variable wherever it makes sense to do so.

After the book has been generated and updated, the variables will have been updated with the values from the map.

For example, if your goal is to include a header/footer variable in your documents that includes the "version" information defined in the DITA map metadata.

1) In your DITA map, include a prodlist/vrmlist/vrm/@version value.
2) In your Book application template file, or in the appropriate component template, add a header/footer variable named "fmx-version" and give it a default value (perhaps "0.00"). Save and close the template file.

   You can also add this variable definition to any "included" FM binary files. The variable value will be updated when the book is generated.

3) In your book-build INI file (or the default book-build INI), enable the ImportAttrsAsVars parameter by setting it to "1".

4) Use the Generate Book from Map command to create a new FM book and chapter files. The pages in the generated output that are based on the template pages which include the variables added in step #2 will contain the updated variable values.

This works because the Book application's XSLT file creates the corresponding attribute (@version in this case) on the fm-ditabook element (in the generated book file). If this isn't working, make sure that your Book application XSLT contains the proper variable definition, and you've followed the instructions correctly.

You should be able to create variables that use any of the <fm-ditabook> attributes listed in the tables in Passing Map-level Metadata to the FM Book. If you want to make use of map metadata not currently supported by the XSLT, just add new XSLT variable processing for a new fm-ditabook attribute and add that attribute definition to the *fmx-book_1.2.dtd* file and to the Book application EDD.

RELATED INFORMATION:

"Adding Map to Book Metadata Mappings" on page 27

Using Templates to Define Alternate TOC Entries

It is possible to support multiple styles of TOC entries for different templates.

By default, the paragraph styles applied to all titles in all generated FM files are the same (title.0, title.1, etc., as defined by the Book structure application). When pulled into a TOC file, they become "title.0TOC", "title.1TOC", and so on. Your TOC generated list template ("*gentpl~toc.fm*") defines the handing for these styles on the "TOC" reference page. If you want to be able to support multiple types of entries, such as "Chapter", "Appendix", and "Part", you'll need to do the following.

Create a custom EDD and template for each TOC type

The component template (*tpl~<mapelemtype>.fm*) will apply alternate formatting to the specific book component, but if you want the underlying paragraph styles to change, you need to create a modified EDD that applies the alternate style, and update the template to support that style. The actual style may look the same, but the name needs to be different so it can be handled separately in the TOC template.

For example, if you want to add support for the appendix element and have your TOC include both Chapter and Appendix labels, you could do the following:

a) Copy the default Book EDD (*book_1.2.edd.fm*) to a new name, perhaps *book_1.2_appendix.edd.fm*.

b) Copy the default Book template (*book_1.2.template.fm*) to a new name, perhaps *book_1.2_appendix.template.fm*.

c) Edit the *book_1.2_appendix.edd.fm* file and change all instances of "title.0" to "title-appendix.0", then save the EDD.

d) In the appendix template (*book_1.2_appendix.template.fm*) rename the "title.0" style to "title-appendix.0" and modify its appearance as needed.

e) Import the EDD into the appendix template, then save and close both files.

f) Copy the appendix template to the proper component template name (from *book_1.2_appendix.template.fm* to *tpl~appendix.fm*, then copy this file into the appropriate component template folders (defined by the BookTemplatesDir parameter in the *ditafmx-book-build.ini* file).

When a book is generated and this template is applied, this will apply the title-appendix.0 style to the top-level heading in the appendix files.

Add the new paragraph style to the GeneratedFile-toc section

For each new TOC entry type, you'll need to add that paragraph style to the GeneratedFile-toc section of the *ditafmx-bookbuild.ini* file. Be sure to update the NumTags parameter to match the number of paragraph tags used to generate the TOC.

Add support for the new TOC styles in the TOC template

In the TOC generated list template ("*gentpl~toc.fm*"), add support for the "title-appendix.0TOC" style on the TOC reference page (it would look just like the title.0TOC style but use the new style name and prepend "Appendix" instead of "Chapter." It should also probably use an alpha numbering style instead of Roman.

You can use the method described above to handle as many alternate TOC entry types that you need in your book.

Including FM Files in a Generated Book

If needed, you can mix FM binary files with DITA-sourced files in the generated book.

There may be times that you'll want to include FrameMaker binary (*.fm*) files in a book that's created by the **Generate Book from Map** command. One particular situation is for the addition of a title page but there may be other uses as well. For example, you may have content that you want to include in a book that has not yet been converted to DITA. With this feature you can build a book that contains some DITA-sourced chapters as well as some conventional, unstructured chapters.

DITA-FMx provides an "Include Files" feature that is implemented through the *ditafmx-bookbuild.ini* file. These files are added to the book at the beginning of the book-build process, just after the DITA files have been aggregated into FM files. To include FM files you'll specify their file name and position in the book (where a position of "1" is the first file in the book). You can also optionally define a "mapelemtype" value to be associated with this file so you can control the pagination properties. To include a title page in the book as the first file and to define this as a type of "titlepage," the following sections should be added to the *ditafmx-bookbuild.ini* file:

```
[IncludeFiles]
1=title.fm

[IncludeFileTypes]
1=titlepage
```

The IncludeFiles section specifies the position and file name, where the file name is relative to the book file. You can specify that multiple files are added to the book, just make sure that the "position" values are always unique, and the files are ordered from the lowest position to the highest. The IncludeFileTypes section defines the "mapelemtype" value for the files at each position. You can use values that match those that are map element types in DITA (such as "chapter" or "appendix") and you can create your own values (such as "titlepage"). Be sure to define the NumberingFirst and NumberingDefault sections as needed to assign numbering and pagination values to these mapelemtype values.

Custom Master Pages

Custom master pages can be applied to specific pages throughout a book to provide alternate page layouts.

By default, the structure application template or the component template will apply one or more master pages to a document based on the settings in the

Master Page Usage dialog (**Format > Page Layout > Master Page Usage**). If you need to apply additional master pages to specific pages you can use the standard FrameMaker custom master page mapping mechanism through the StructMasterPageMaps table Reference page. This mapping table lets you associate master pages to paragraph or element tags and define very detailed mapping contexts. For information on this feature, refer to the FrameMaker documentation.

In order for the StructMasterPageMaps mapping table to be applied to the book components, you must select the **Apply Master Pages** option in the Book Build Settings dialog or set the BookBuildOverrides/UpdateBook parameter to “2” in the book-build INI file. Setting this value to “1” updates the book, but does not apply the mapping table.

If you’d like to apply the occasional master page without needing to set up this mapping table, insert an fmx-masterpage marker with the marker text set to the master page name to apply. Additionally, you must set the BookBuildOverrides/AssignMasterPagesFromMarkers parameter to “1” in the book-build INI file.

You can use an unstructured fmx-masterpage marker, which is stored in the DITA file as a processing instruction, or you can use the <fm-data-marker> element (which is stored as a DITA <data> element). If you use the <fm-data-marker> element, you’ll need to create the fmx-masterpage marker definition in the Topic and Book structure application templates (and potentially any component templates that are used).

NOTE: *When using the <fm-data-marker> element as the fmx-masterpage marker, do not use the “Conditionalize data and data-about" option along with "Hide Conditionalized Content" option in the DITA-FMx Options dialog.*

Setting Up PDF Bookmarks

It can be tricky to get the PDF bookmarks set up properly, but it’s worth the effort.

When generating a PDF from a book, the default list of paragraph styles used to create the bookmarks contains all paragraph styles that exist in all files in the book, but the include/exclude settings are defined by those in the first file in the book. If you want the bookmark list to be ready to use (without needing to move items over to the exclude list every time), you must make sure that the first file in the book contains all of the styles in all files, as well as being set up with the proper include/exclude settings.

Use the **Merge Para Tags** command to copy the paragraph tags from all other templates in a book.

If the first file in the book is generated from a DITA topic, you’ll need to set up the app’s template file as described below. If the first file in the book is a generated file, you'll need to perform this setup on the generated list template. Or if the first file is a title page, that file needs to be set up accordingly.

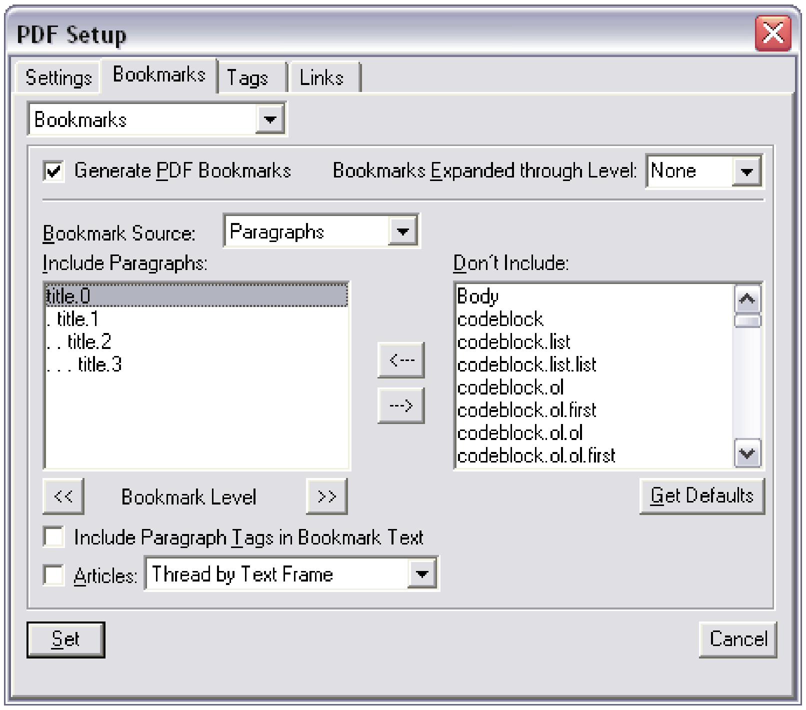
1) Import all paragraph styles from all possible documents into the file. It's OK to have more styles than are used, but any that are in files other than the "first" file will show up in the "included" bookmark list.

2) Open the file and run the **Format** > **Document** > **PDF Setup** command.

3) In the PDF Setup dialog, on the **Bookmarks** tab, select **Generate PDF Bookmarks** and "Paragraphs" as the **Bookmark Source**, then set up the "Include" and "Don't Include" styles and adjust the bookmark level as needed.

4) Set any other properties as needed in the PDF Setup dialog, then choose the **Set** button. This saves the PDF setup data to the file.

NOTE: This process will only work for FM binary files, you cannot save this setup data to an XML file.



Bookmarks

☑ Generate PDF Bookmarks   Bookmarks Expanded through Level: None

Bookmark Source: Paragraphs

Include Paragraphs:
title.0
. title.1
. . title.2
. . . title.3

Don't Include:
Body
codeblock
codeblock.list
codeblock.list.list
codeblock.ol
codeblock.ol.first
codeblock.ol.ol
codeblock.ol.ol.first

☐ Include Paragraph Tags in Bookmark Text

☐ Articles: Thread by Text Frame

Figure 1-15: PDF Setup dialog box

TIP: To quickly move all paragraph tags from the left list to the right list, hold down the SHIFT key and click the '->' (right arrow) button.

Related information:

"Merge Para Tags" on page 13

2 Installation and Setup

Detailed information about installing DITA-FMx.

DITA-FMx supports DITA 1.2 (and earlier versions) on FrameMaker versions 7.2, 8, 9, 10, 11, 12, 2015, 2017, 2019, and 2020, and can be downloaded from www.leximation.com/dita-fmx/.

DITA-FMx is made up of two plugin components, one that provides import/export processing and one that provides general authoring support. DITA-FMx also provides three structure applications (DTD, EDD, template, and read/write rules files), two for DITA topic and map authoring and one for book publishing. The EDD and rules files have been set up to allow proper interaction with DITA-FMx. If you want to build your own DITA structure applications, you should clone these files and modify them as needed. For more information see Developing Custom Structure Applications.

As of DITA-FMx 2.0.06, support for authoring and publishing of Lightweight DITA (pre-release) is available.

We've posted a video of the complete installation process. Please take a moment to watch this video:
http://blog.leximation.com/2010/03/installing-dita-fmx-1-1/

If you have any installation problems or questions, please contact us at <ditafmx-help AT leximation DOT comditafmx-help@leximation.com>.

RELATED INFORMATION:

"Using DITA-FMx" on page 1
"DITA-FMx Commands" on page 1
"Extending DITA-FMx" on page 1

Before Running the Installer

Be sure to address all appropriate pre-installation tasks before running the installer application.

IMPORTANT: *If installing DITA-FMx on FM 2017, you must have FrameMaker version 14.0.2 or later. Please update to that patch level before installing DITA-FMx.*

RELATED INFORMATION:

"Run the Installer Application" on page 8

Installing DITA-FMx for the First Time

As of DITA-FMx 2.0.07, the installation application modifies the maker.ini file to disable the existing DITA support; on earlier releases, you needed to do this manually.

DITA-FMx cannot coexist with the default DITA support. An error message will display to remind you of this if the *maker.ini* file has been edited to enable the default DITA support.

NOTE: *If installing on FrameMaker 7.2, read Installing DITA-FMx on FrameMaker 7.2 instead of this topic. For all other version of FrameMaker continue reading below.*

The information in the following bullet item is provided for your information only; there is no action that is required.

- After installation, the *maker.ini* file (in the main FrameMaker program folder) will have been modified to comment out the default DITA support. In the APIClients section, you should see lines that start with **ditafm**, **ditafm_app**, **ditabook** (FM8 only), and possibly **ditaOpenToolkit** or **DITA OT** (in FM12/2015/2017/2019/2020 or if you've installed the DITA OpenToolkit connector).

  In FM12/2015/2017/2019/2020, you should see lines that look like the following:

```
;ditafm=Standard, Translation client for DITA, fminit\ditafm.dll, structured
;ditafm_app=Standard, Translation client for DITA, fminit\ditafm_app.dll, structured
;Dita OT=Standard, DITA OT client for DITA, fminit\openToolkit.dll, structured
```

NOTE: *On Windows Vista, 7, 8, or 10, in order to edit the maker.ini file you may need to run Notepad (or whatever text editor you are using), "As Administrator". This does not mean that you have Administrator permis-*

sions, but that you right-click the program icon and choose "Run As Administrator."

FM 10-12 On FrameMaker 10 and later, you must restart FrameMaker after initial setup to ensure that the proper structure applications have been registered in the default FrameMaker ditafm.ini file. Due to changes in these FrameMaker versions, the structure applications are stored in both the ditafmx.ini and the default ditafm.ini files.

NOTE: *If you're interested in switching between FM-DITA and DITA-FMx, refer to the topic Switching Between DITA-FMx and FM-DITA.*

You may now proceed to Run the Installer Application.

RELATED INFORMATION:

"Switching Between DITA-FMx and FM-DITA" on page 49
"Run the Installer Application" on page 8
"Special Instructions for FM12 and Earlier" on page 12

Migrating from FM-DITA to DITA-FMx

Tips for migrating from default FrameMaker DITA to DITA-FMx.

In general, there should be minimal effort required to migrate from the default DITA support in FrameMaker (FM-DITA) to DITA-FMx. Because both DITA authoring environments create valid DITA files, you can open and edit files created by the other environment just as you can open and edit files created in different authoring tools such as XMetaL or Oxygen.

If you've customized the default FrameMaker DITA structure applications and want to continue using those customizations in DITA-FMx, there may be some work required to migrate the apps since DITA-FMx provides features that do not exist in FM-DITA. For a complete listing and description of the DITA-FMx structure application requirements, refer to the DITA-FMx Specific Element Definitions topic.

If you want to just "give it a try" to see what breaks, that's fine (you're not going to hurt anything). The only thing that's absolutely required is that you change the "Use API Client" node in the application definition so it uses the "ditafmx-_app" client name instead of "ditafm_app." You'll need to do this for any structure application that you want to use with DITA-FMx. (If you try to use these apps without changing the UseApiClient node, you will get the following error message when opening a DITA file: "Translator client (ditafm_app) was not found.") Once this is done, change the application names in the DITA Options dialog, and give it a whirl.

If you're migrating from an earlier version of FM-DITA, which may support DITA 1.0 or 1.1, keep in mind that because DITA-FMx 2.0 supports DITA 1.2, there will be significant differences in the EDDs. You can continue to use older

DITA apps in DITA-FMx 2.0 as long as you provide certain element definitions (described below).

One significant difference between FM-DITA and DITA-FMx (since FM8) is that DITA-FMx relies on a separate “Book” structure application for publishing. The “Topic” application is for authoring topics, the “Map” application is for authoring maps, and the “Book” application is for publishing. While this does add a bit more effort to maintain one more structure application, it provides far greater control over your publishing process. Also keep in mind that while you may be tempted to do so, there’s really no reason to customize the “Map” or “Topic” apps (unless you’re adding specialized elements). You shouldn’t need to see the published view of the content while authoring, so you really just need to customize the Book app to align with your publishing styles.

The following items are significant migration points, refer to the DITA-FMx Specific Element Definitions topic for a complete list of elements specific to DITA-FMx.

The <fm-indexterm> element is required (if you use an index).

In the DITA-FMx Topic and Book applications the <fm-indexterm> element is required to exist and be defined as a marker element type. You should be able to rename the FM-DITA <indexterm> to <fm-indexterm> since by default the FM-DITA <indexterm> is a Marker element. You’ll also need to rename all references to <indexterm> in the general rule of other element definitions. If you want to be able to use the DITA-FMx feature that lets you switch between an <fm-indexterm> Marker and an <indexterm> Container, you’ll need to create an <indexterm> element that is defined as a Container. This should be added to all general rules at the same location as the <fm-indexterm>.

For more information, refer to the indexterm and fm-indexterm topic.

The topic referencing label in Maps.

The DITA-FMx Map application uses the <fm-reflabel> element to provide a clickable label for navigation in a map or bookmap. In DITA-FMx 1.0 as well as in FM-DITA, this element is named <fm-topicreflabel>. Because bookmaps can contains topic referencing elements other than <topicref>, it was decided that this element name should be changed.

For more information, refer to the fm-reflabel and Topic References topic.

Special Book application elements.

The DITA-FMx Book application includes three required elements, <fm-ditabook>, <fm-ditafile>, and <fm-bookcomponent>. The <fm-ditabook> element makes use of href and title attributes, and the <fm-ditafile> element uses the href and mapelemtype attributes. The Book application will also use the <fm-tabletitle> element if you enable the “Move table/title” book-building option and the <fm-figuretitle> element if you use the “Move fig/title” option. If you plan to take advantage of the enhanced DITA-FMx book-building functionality, you should make sure your Book application includes these elements.

For more information, refer to the DITA-FMx Specific Book Application Elements topic.

The simpletable-based table structures.

All of the DITA-FMx applications make use of <simpletable>-based tables. In Topic and Book applications are <simpletable>, <choicetable>, and <properties> (table), and in the Map application is the <reltable>. Because tables within FrameMaker require a slightly different structure than that required by the DITA specification, additional “fm-*” elements are added as wrappers to the “head” and “row” (<sthead> and <strow> in <simpletable>) elements. DITA-FMx uses a slightly different structure than FM-DITA. The base table name is the root element, which contains “fm-*TABLENAME*head” and “fm-*TABLENAME*body” elements, which in turn contain the standard head and row elements.

For more information, refer to the simpletable-Based Tables topic.

Related information:

"Switching Between DITA-FMx and FM-DITA" on page 49

Installing DITA-FMx on FrameMaker 7.2

If installing on FrameMaker 7.2 and other DITA support is installed, you will need to disable that DITA support.

If you are upgrading from the Adobe DITA App Pack or from the first beta version of DITA-FMx (v.0.01), you may need to modify the *maker.ini* file or disable existing client plugins. If you’re new to DITA authoring or haven’t installed either of these plugins, proceed to Run the Installer Application.

IMPORTANT: *The structure applications provided with DITA-FMx 2.0 are in FM8 format. FM7.2 users will need to open them in FM8 and downsave to FM7 format. If you’re unable to do this, contact Leximation.*

In order to eliminate any possible conflicts with previous versions of the DITA authoring plugins, check the following:

- Verify that there are no “ditafm” or “ditafmx” plugin DLLs in the *FrameMaker/fminit/Plugins* folder (or any subfolders). Files to look for are: *ditafm.dll*, *ditafm_app_72.dll*, *fm-ditabook.dll*, *ditafmx_72.dll*, and *ditafmx_app_72.dll*. If any of these files are found, move them to a backup location outside of the *Plugins* folder.

This only applies to files in the “Plugins” folder; files found in the *\fminit\ditafm* folder can be left alone.

IMPORTANT: Do not rename the DLLs or move them to a sub-folder of the Plugins folder. You must completely move them to an alternate location. If the files exist (under any name) within or below the Plugins folder, FrameMaker will still load them.

- Open the *maker.ini* file and locate the APIClients section. Look for lines that start with “ditafm” or “ditafm_app” and delete or comment them out (by adding a semicolon at the start of the line). Lines that start with “ditafmx” or “ditafmx_app” will be updated by the installer and do not need to be commented out.

You may now proceed to Run the Installer Application.

Upgrading from DITA-FMx 1.0 or 1.1

Tips for a painless migration from an earlier version of DITA-FMx.

DITA-FMx 2.0 should handle your existing DITA-FMx 1.1 structure applications without any problems, especially if you’re upgrading from the latest release (1.1.18). If you are using an older app, some of the new features that rely on specific structure application settings may not work. But you should be able to install the update and continue working, and expect the same level of functionality that you’re used to. When you are ready to migrate the new features into your structure application, you should review the DITA-FMx Specific Element Definitions topic.

The default structure application provided with DITA-FMx 2.0 supports the DITA 1.2 model. You should be able to use this app for DITA files created with earlier DITA models, but you can also continue to use your existing apps if you prefer.

TIP: The changes made to the structure applications are documented in the “Structure Application Updates” sections of the Revision History for each update. It may be useful to review all of these changes from your version when upgrading a custom structure application.

Because DITA-FMx 2.0 installs into a new directory structure (*FrameMaker\DITA-FMx-2*), it will not overwrite your DITA-FMx installation. After running the installer, run FrameMaker and you’ll be prompted to install the new DITA 1.2 structure applications. These are also installed to a new directory structure so will not overwrite your existing application files.

This allows you to maintain both DITA-FMx 1.1 and 2.0 installations side-by-side. To switch from one to the other, see Switching Between DITA-FMx 1.1 and 2.0.

After installing DITA-FMx 2.0, you may want to migrate some of the settings in the 1.1 *ditafmx.ini* file. There have been some changes to this INI file, but some of the settings will be the same.

RELATED INFORMATION:

"Installing a DITA-FMx Update" on page 7
"DITA-FMx Specific Element Definitions" on page 29

Installing a DITA-FMx Update

Basic options for upgrading your DITA-FMx installation to the latest point release.

To reinstall or update DITA-FMx over an existing installation, you've got two basic options.

Quick and simple, but unable to revert to the previous version

When installing an interim update, or you don't care about being able to roll back to the previous version, this approach is for you.

Just run the installer. No backup or uninstall is required. The installer will overwrite all like-named files in the *DITA-FMx-2* folder. It will not overwrite the *ditafmx.ini* file, so your settings will not be lost.

Fresh install, and able to revert to the previous version

If you'd like to get a "fresh" install, you should rename the existing *DITA-FMx-2* folder (perhaps to something like *DITA-FMx-2.<oldversion>*) before running the new installer. This will give you a completely new set of files, and will let you compare the new files to the old files and allow you to "roll back" to a previous version if needed. If you choose this approach, you will need to manually migrate any manually entered settings in the *ditafmx.ini* file. In particular, you'll need to migrate the info in the Registration section. Just copy and paste the entire Registration section into the new *ditafmx.ini* file after it is created, then restart FrameMaker. (Otherwise, you'll need to enter your registration information and auth code in order to initialize the application.)

After installing the update, you may want to copy the new structure applications to the *FrameMaker\Structure\xml\* folder so you can take advantage of any changes that have been made. The DITA-FMx installer does not install the new structure applications, they are provided in a ZIP file in the *DITA-FMx-2* folder. If you're using your own custom applications, you should review the DITA-FMx Revision History, so you can integrate any structure application updates. In particular, you should consider using the updated XSLT scripts for the Book application to take advantage of any fixes to the book-build processing.

NOTE: *If you're installing an update on Windows Vista, 7, 8, or 10, keep in mind that the user-editable INI files are in the "app data" area not in the Program Files area.*

IMPORTANT: *If you're installing an update on FM12, due to a core problem in FrameMaker, you must manually delete the ModalDialogPosition.ini file to allow updated DITA-FMx dialogs to display at their proper size. This INI file can be found in "%appdata%\Adobe\FrameMaker\12".*

RELATED INFORMATION:

"Special Instructions for FM12 and Earlier" on page 12

Run the Installer Application

Extract the EXE from the ZIP archive and run it.

The DITA-FMx installer application is provided in a ZIP archive. Extract the executable from the archive and run it. The installer extracts and copies the required files to the *FrameMaker\DITA-FMx-2* folder.

As part of the installation process, the installer modifies the *maker.ini* file. If you feel it is necessary, you may want to make a backup of this file before running the installer. If you are installing an update, see Installing a DITA-FMx Update.

The following files are installed:

- ***ditafmx***_<fmver>.dll - Authoring and publishing support plugin DLL.
- ***ditafmx***_<fmver>_app.dll - Import/export client DLL.
- ***pt_updates.dll*** - The DLL that provides web access for the "check for updates" command.
- ***ditafmx***.chm - DITA-FMx User Guide online Help file.
- ***ditafmx-ref-1.1.chm*** - DITA 1.1 Reference online Help file (for element-based context sensitive authoring help). Includes the special DITA-FMx elements.
- ***ditafmx-ref-1.2.chm*** - DITA 1.2 Reference online Help file (for element-based context sensitive authoring help). Includes the special DITA-FMx elements.

- ***DITA-FMx_Help_Source.zip*** - Source for the user documentation.
- ***DITA-FMx_apps.zip*** - Structure application files for DITA 1.2.
- ***LwDITA-FMx_apps.zip*** - Structure application files for Lightweight DITA.
- ***ditafmx-ant.xml*** - Ant script that provides targets for the Current File option of the Generate Output command.
- ***PROJECT.xml*** - Sample Ant script for the Selected Target option of the Generate Output command.
- ***ditafmx-bookbuild.ini*** - The default book-build INI used by the Generate Book from Map (Book Build) process. Copy this file into the folder that contains the generated book file to customize the build process for that book. If no book-build INI file is found in the output folder, the settings in this default file are used.
- ***filtergroups.ini*** - The default INI file used by the filtering groups feature of the Set Attributes command. Copy this file into the folder that contains the topic files to create project-specific filtering groups. If no *filter-groups.ini* file is found in the topic folder, the settings in this default file are used.
- ***missing-images*** folder - Contains files named *missing-image-break.<ext>* and *missing-image-inline.<ext>*. Files are provided for all conceivable file types. If you a re using files of a type not provided, create them from a type provided. The *_README.txt* file in this folder explains why they are needed.
- ***ditafmx-zip.exe*** - A ZIP archive creation utility provided by Info-ZIP. Used by the Create Archive command.
- ***ditafmx-unzip.exe*** - A ZIP archive extraction utility provided by Info-ZIP. Used by the automated structure application installation process.

After extracting the new files, the installer updates the APIClients section of the *maker.ini* file with references to the plugin DLLs. The following lines are added (the numbers will vary based on the FrameMaker version: “160” is FM2020, “150” is FM2019, “140” is FM2017, “130” is FM2015, “120” is FM12, and so on):

```
ditafmx=Standard, ditafmx, DITA-FMx-2\ditafmx_160.dll, structured
ditafmx_app=Standard, ditafmx_app, DITA-FMx-2\ditafmx_160_app.dll, structured
```

IMPORTANT: *As of DITA-FMx 2.0.07, the installer application modifies the maker.ini file to comment out references to the default DITA plugins; you no longer need to do this manually.*

NOTE: *On Windows Vista, 7, 8, or 10, when you run FrameMaker, the user data files (like the ditafmx.ini file) are created in a DITA-FMx-2 folder in the applica-*

tion data area (typically C:\Users\USER-NAME\AppData\Roaming\Adobe\FrameMaker\16\DITA-FMx-2). Any "global" data files that you create (like filtergroups.ini and ditafmx-book-build.ini), that the documentation indicates should be in the FrameMaker/DITA-FMx folder should be created in this location as well. The ***DITA-FMx > Open DITA-FMx Folder*** *command is useful for accessing this folder.*

Once the installation has completed, you need to set up the structure applications before you can create and edit DITA files. Continue to DITA Structure Applications.

DITA Structure Applications

You can use the DITA structure applications provided with DITA-FMx, those provided by Adobe, or your own custom applications. If you're using apps other than the DITA-FMx apps, you'll need to make some minor adjustments.

In order for FrameMaker to properly open and save DITA files, you must have structure applications that support the authoring of topic and map files. DITA-FMx provides both of these applications as well as a structure application for book processing.

Using the default DITA-FMx structure applications is the easiest way to get started, because those applications will support all of the advanced features provided by DITA-FMx.

NOTE: *If you're installing on a system where the user does not have "admin" permissions, you'll want to install the structure applications in a more accessible location. For details, see Alternate Location for the Structure Applications.*

IMPORTANT: *The structure applications provided with DITA-FMx 2.0 are in FM8 format. This works fine for FM8 and later, but if you're using FM7.2, you'll need to open them in FM8 and downsave to FM7 format. If you're unable to do this, contact Leximation.*

If you are already working with DITA structure applications, and want to continue using them, you'll need to do the following:

- Edit the structure application definition for each of your structure applications. Change the **Use API Client** node so it references the client **ditafmx_app** instead of **ditafm_app**. This is required; if you do not do this, you'll get an error message.

- Add new element definitions that support special features provided by DITA-FMx. For details see DITA-FMx Specific Element Definitions. This is optional, but may limit the available functionality.
- Create a special “Book” application for publishing. DITA-FMx relies on a structure application for Book publishing. It is possible that your current “Topic” application will do the job.

If you want to customize the appearance of default DITA-FMx structure applications, you should clone the provided structure application folders and modify the cloned versions. For additional information, refer to Cloning the Default Structure Applications.

After installing DITA-FMx, when you run FrameMaker you should see a message asking to install the default structure applications. Unless you’re doing something special, you should choose Yes to this message. To manually install the structure applications or if you’d like to understand what this automated process is doing, read the topic titled Manual Installation of the Structure Applications.

IMPORTANT: *On Windows Vista, 7, 8, or 10, the automated structure application installation process may fail unless FrameMaker is run “As Administrator.” Exit FrameMaker, then right-click the program icon and choose “Run As Administrator.”*

After the structure applications have been installed, you may want to take a look at the structure application definitions so you will better understand how they work. Just choose **Structure Tools > Edit Application Definitions** to open the structure application definitions file. You should see the three DITA-FMx structure application definitions at the top of the file. These are added as text insets to make it easier to manage. If you need to modify these entries, you’ll ned to open the source file by double-clicking the inset and choosing the Open button. When you’re done reviewing the structure application definitions file, just close the file. You’ll be prompted to save when closing. Unless you’ve made intentional changes, there is no need to save.

Special Instructions for FM12 and Earlier

With the FM13.0.1 update, the default XSLT processor changed to SAXON. This requires some extra work when setting up the Book structure application for installations on FM12 and earlier.

The DITA-FMx Book structure application makes use of an XSLT import script (*bookmap2fmbook.xsl*) to aggregate the topic files in the map. There are two versions of this XSLT script, one that requires the XALAN processor and one that requires SAXON. Because the default XSLT processor since FM 2015 is SAXON, that’s the version that is set up as the default.

If you're installing DITA-FMx on FM12 or earlier, you'll need to copy the XALAN version of this script over the default XSLT file because XALAN is the default XSLT processor on those versions of FrameMaker.

NOTE: *Technically, if you're installing on FM 2015 with no updates installed, you'll also need to use the XALAN version, but we assume that you'll be installing the FM updates, so as long as the FM version is 13.0.1 or greater, you should be fine with the default SAXON version.*

It is possible to switch the default XSLT processor on FM12 or FM11 to SAXON, so you may want to do that instead of using the XALAN version of the script. Also, it is possible to change the default processor to XALAN on FM 2015 and later.

To change the default XSLT processor on FM11 and later:

1) Open the *maker.ini* file in a text editor and locate the XSLTProcessors section.
2) Move the "**, Default**" parameter (the word "Default" and the leading comma) to the line that specifies the processor type you want to be the default.
3) Save and close the *maker.ini* file then restart FrameMaker.

Once you've set the default XSLT processor, locate the Book structure application folder, and you should see three similarly named XSLT files:

- *bookmap2fmbook.xsl*
- *bookmap2fmbook-SAXON.xsl*
- *bookmap2fmbook-XALAN.xsl*

Delete the *bookmap2fmbook.xsl* file, then copy the XSLT file that indicates the processor you want to use (SAXON or XALAN), to the file name *bookmap2fmbook.xsl*. Using this technique, you can switch from SAXON to XALAN as needed.

If you have a custom structure application, you'll need to make sure it's using the proper XSLT file for the default XSLT processor.

Manual Installation of the Structure Applications

Perform a manual installation of the structure applications or gain an understanding of what needs to be done.

The following steps describe the installation of the structure applications provided with DITA-FMx.

NOTE: If you're installing on a system where the user does not have "admin" permissions, you'll want to install the structure applications in a more accessible location. For details, see Alternate Location for the Structure Applications.

TIP: We've posted a video of the complete DITA-FMx installation process. If you haven't watched this video you may want to take a moment to do so: http://blog.leximation.com/2010/03/installing-dita-fmx-1-1/

1) Make a backup copy of your current structure application definitions file (typically found at *FrameMaker\Structure\structapps.fm* for FM7.2 and FM8, or <*user appdata*>*\Adobe\FrameMaker\*<*version*> for FM9 and later). Store this file in a safe location before making modifications.

2) Extract the contents of the *DITA-FMx_apps.zip* file to the *FrameMaker\Structure\xml* folder. This will create a folder named *DITA-FMx_1.2* that contains folders named *Book*, *dtd-fmx*, *Map*, and *Topic*. These folders contains the three structure application as well as the DITA 1.2 DTD files used by the applications.

3) Start FrameMaker and open the structure application definitions file (**Structure Tools** > **Edit Application Definitions**). (If running FM9 or later, this opens the *structapps.fm* file in your "USERNAME\AppData" area, not the one in the FrameMaker program files area).

4) Open the Structure View window (for FM7.2 or FM8 click the button on the upper right of the document window, in FM9 and later choose **Structure Tools** > **Structure View**). In the structure application definitions file place the insertion point just after the Version element. When the insertion point is in the right location, you'll see a black triangle pointing to the right in the Structure View window (see the following image).

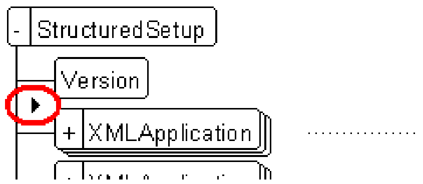


Figure 2-1: Structure View window insertion point

5) Choose **File** > **Import** > **File**, then navigate to the *structapps-stub_topic_1.2.fm* file in the *DITA-FMx_1.2\Topic* folder created in step 2. Select the Import by Reference option and choose the Import button. In the next dialog accept the defaults and choose Import.

NOTE: If you see red dotted lines in the Structure View window after importing the "stub" file, it has been inserted into the wrong location. Delete the inset and try again. Make sure that the black triangle is placed as shown in Figure 1-1.

6) Repeat step 4 for the “structapps-stub” files in the *DITA-FMx_1.2\Map* and *DITA-FMx_1.2\Book* folders.

7) Save the file, then choose **Structure Tools** > **Read Application Definitions**.

8) Close the file and exit FrameMaker.

9) Restart FrameMaker and run the **DITA-FMx** > **Options** command and select **DITA-FMx-Topic-1.2** for the DITA Topic Application, **DITA-FMx-Map-1.2** for the DITA Map Application, and **DITA-FMx-Book-1.2** for the DITA Book Application.

NOTE: If your folder structure is non-standard, you may need to modify the paths specified in the “structapps-stub” files to match the file paths on your system. To do this, just open the structure application definitions file and double-click each application’s text inset. In the dialog that displays, choose the Open Source button and make the changes in the “stub” file. When you have finished editing the stub file, save and close that file, then double-click the text inset again and choose the Update Now button.

If you plan to make use of the DITA Open Toolkit for generating output, see Publishing with the DITA Open Toolkit.

Alternate Location for the Structure Applications

On systems where the author does not have admin privileges, it may make sense to install the structure applications to an alternate location that is more accessible.

The structure applications should be installed to a location that’s writable for the end user. This is to allow for modifications to be made as needed.

The following steps describe an option for installing the structure applications in a more accessible location.

1) Select a writable location for the structure applications.

   It doesn’t really matter where you choose, just some place that is writable by the end user. These instructions will use the path *C:\FM-apps*.

2) Copy the app files into the selected folder.

   Extract the contents of the *DITA-FMx_apps.zip* file (found in the *Program Files\Adobe\FrameMaker\DITA-FMx_2* folder) into the *FM-apps* folder. This will create a folder named *DITA-FMx_1.2* which contains three folders, *Topic*, *Map*, and *Book*. Each of these three folders contain a “structapps-stub” file specific to that structure application (*struct-*

apps-stub_topic_1.2.fm, *structapps-stub_map_1.2.fm*, and *structapps-stub_book_1.2.fm*).

3) Modify the path variables in each of the “structapps-stub” files.

Open the *structapps-stub_topic_1.2.fm* file in FrameMaker. You’ll see a number of file references, starting with the text “$STRUCTDIR,” these are FrameMaker variables. The goal is to change these to reference the new location for the files. Double-click the variable, then click the “edit” button in the Variables panel or dialog (this functionality will vary based on the FrameMaker version). Change the definition of this variable from “$STRUCTDIR\\xml\\DITA-FMx_1.2\\” to “C:\\FM-apps\\DITA-FMx_1.2\\” (make the actual value match the real location). Be sure to use double backslashes and to include the trailing slashes.

Choose “Change” and complete the process of modifying the variable. You should see that all instances of this variable have updated to use the new path.

Do the same thing for the Map and Book “structapps-stub” files.

4) Follow the instructions for manually installing the structure applications.

See, Manual Installation of the Structure Applications. Start with step #3, and in step #5 navigate to the “structapps-stub” files in the folders you just set up.

IMPORTANT: *If you have already installed the structure applications to the default location, you should remove the entries in the structure application definitions file that reference those applications. You can only have one application definition for a named application.*

Description of the Structure Application Files

The structure application files provided with DITA-FMx do more than your typical DITA structure applications.

After installation of the DITA-FMx structure applications, the *FrameMaker\Structure\xml\DITA-FMx_1.2* folder contains a number of files, some are specific to the structure applications, and some are used for other purposes.

IMPORTANT: *The DITA-FMx structure applications are provided in FM8 format. This should work fine for FM8 and later versions of FrameMaker. FM7.2 users willl need to downsave to the FM7 format. If you are unable to do this, contact Leximation.*

DITA-FMx_1.2

map-from-outline_template.fm - Template file used by the **New DITA File > New Map from Outline** command.

DITA-FMx_1.2\Book\

book_1.2.edd.fm - The Book application EDD. Defines the data model structure and provides the element definitions. Remember to import this into the template (below) if you make and changes to the EDD.

book_1.2.rules.txt - The Book application read/write rules file.

book_1.2.template.fm - The Book application template file. Defines the page layout and formatting as well as the character and paragraph styles used by the EDD.

bookmap2fmbook.xsl - The Book application XSL import file. Controls the process that aggregates all of the topic files into the book and chapter FM files.

expandOrig.xsl - A support file for the Book XSL import process. Performs initial aggregation of all topic files before passing control off to the *bookmap2fmbook.xml* file.

structapps-stub_book_1.2.fm - The Book application definition stub file. Inserted into the structure application definitions file.

structapps-stub_book_1.2_13.0.1.fm - The Book application definition stub file for FM13.0.1 and later. Inserted into the structure application definitions file.

DITA-FMx_1.2\Book\component-templates\

gentpl~indexlist.fm - A sample template for generating an Index (indexlist). Referenced by the *ditafmx-bookbuild.ini* file.

gentpl~toc.fm - A sample template for generating a TOC (toc). Referenced by the *ditafmx-bookbuild.ini* file.

tpl~appendix.fm - A sample template for an Appendix map element. Referenced by the *ditafmx-bookbuild.ini* file.

tpl~chapter.fm - A sample template for a Chapter map element. Referenced by the *ditafmx-bookbuild.ini* file. The default Chapter template is actually just a copy (and rename) of the default Book template. It can be convenient to use a Chapter template to make adjustments to the output without modifying the core Book template.

tpl~part.fm - A sample template for a Part map element. Referenced by the *ditafmx-bookbuild.ini* file.

tpl~topicref.fm - A sample template for a topicref map element. Referenced by the *ditafmx-bookbuild.ini* file. The default topicref template is actually just a copy (and rename) of the default Book template (same as the Chapter template). It can be convenient to use a topicref template to make adjustments to the output without modifying the core Book template.

tpl~topicref-HAZALT.fm - An alternate version of the *tpl~topicref.fm* file that defines the hazard table objects for a full-width header table. For details, see Hazard Statement Publishing Options.

DITA-FMx_1.2\common\

Because the EDD structure is mostly identical between the Topic and Book applications, we have set up groups of element definitions as text insets. The *common* folder contains these files. A book file is provided as an easy way to search through the files.

DITA-FMx_1.2\dtd-fmx\

In order to speed up processing of the complex DITA DTD structure, single-file versions of the DITA DTDs have been created for use with the default DITA-FMx structure applications. This is particularly important if you are using FM7.2 or FM8, but is beneficial for all versions of FM. You are free to reference the standard versions of the DITA DTDs if preferred.

fmx-ditabase_1.2.dtd - Used by the default Topic application.

fmx-map_1.2.dtd - Used by the default Map application.

fmx-learning_1.2.dtd - Used by the default Learning application.

fmx-book_1.2.dtd - Used by the default Book application.

DITA-FMx_1.2\Map\

map_1.2.edd.fm - The Map application EDD. Defines the data model structure and provides the element definitions. Remember to import this into the template (below) if you make and changes to the EDD.

map_1.2.rules.txt - The Map application read/write rules file.

map_1.2.template.fm - The Map application template file. Defines the page layout and formatting as well as the character and paragraph styles used by the EDD.

structapps-stub_map_1.2.fm - The Map application definition stub file. Inserted into the structure application definitions file.

DITA-FMx_1.2\Topic\

topic_1.2.edd.fm - The Topic application EDD. Defines the data model structure and provides the element definitions. Remember to import this into the template (below) if you make and changes to the EDD.

topic_1.2.rules.txt - The Topic application read/write rules file.

topic_1.2.template.fm - The Topic application template file. Defines the page layout and formatting as well as the character and paragraph styles used by the EDD.

structapps-stub_topic_1.2.fm - The Topic application definition stub file. Inserted into the structure application definitions file.

DITA-FMx_1.2\Topic\element-templates\

new~reference~simple ref.fm - A sample “reference” element template named “simple ref”.

new~task~simple task.fm - A sample “task” element template named “simple task”.

new~topic~simple ref.fm - A sample “topic” element template named “simple topic”.

Initializing DITA-FMx

Request your authorization code and verify that everything is set up properly.

After you’ve run the DITA-FMx installer and set up the structure applications, there are a couple things that you may need to do before you’re ready to use DITA-FMx.

If you are new to DITA-FMx, you’ll need to request an authorization code. For a 30-day trial authorization code, choose the **DITA-FMx > Unregistered. Try Now?** menu item (near the bottom of the menu). Follow the on-screen instructions and enter your email address on leximation.com. Your trial authorization code will be sent to the email address you provide. If your trial authorization code has expired and you need more time, just contact us by email and we’ll be happy to provide an extension.

If you’ve purchased DITA-FMx, your permanent authorization code is available in your Tool Administration area on leximation.com. Just log on to leximation.com, then click the “Tool Administration” link in the lower left of any page. On this page you should see an entry for DITA-FMx with a “request auth code” link. Click that link and your auth code will be sent to the email address registered to your leximation.com account.

Once you get your authorization code (trial or permanent), select the **DITA-FMx > Enter Authorization Code** menu item and provide the information requested. This should unlock the items in the DITA-FMx menu.

Now you should verify that the proper structure applications are selected in the DITA Options dialog. Choose **DITA-FMx > DITA Options**. The Topic, Map, and Book applications should list those that you want to use. If you're using the default DITA-FMx applications that would be DITA-FMx-Topic-1.2, DITA-FMx-Map-1.2, and DITA-FMx-Book-1.2. If these applications aren't available in the drop-lists, you should go back and review the DITA Structure Applications topic; if these applications have been properly installed, their names should be available in the drop-lists in the Options dialog.

You may also want to set other options at this time. Use the Help button in the Options dialog to read about the many options available in DITA-FMx.

NOTE: *Updates for DITA-FMx 2.0 are provided on a "maintenance license." When you purchase DITA-FMx you get the first year of updates at no additional cost. If you want to continue to receive updates for another year, you'll need to pay for another year of maintenance. If you decide not to extend your license, you can continue to use the update that was available during your valid license period. You can always check the state of your maintenance license by choosing* ***DITA-FMx > Registered to (YOUR NAME)****.*

Advanced Installation/Customization Issues

For those people who want to customize the structure applications or modify the functionality of the authoring environment. If you are new to DITA, and have completed the previous installation steps, you should be all ready to use DITA-FMx for creating and authoring DITA XML files.

Developing Custom Structure Applications

Once you're comfortable with the features in DITA-FMx, you'll surely want to create your own custom structure applications that apply your house style to your DITA documents.

In FrameMaker, all of the formatting and page layout applied to XML files is done by the selected structure application. A structure application (also called a "structure app" or just "app") is made up of the following components:

- **Template file** – contains the page layout, paragraph/character formatting, other object formatting (such as cross-ref and variable definitions), and any document-specific interface defaults (such as the visibility of element boundaries).

- **EDD (element definition document)** – defines the structure of the elements in the data model, and the context rules for formatting those elements.
- **DTD (document type definition) files** – a set of markup declarations that define a document type for SGML-based markup languages (XML, SGML, HTML). This includes files of type DTD, ENT, MOD, and others that define the structure of a data model.
- **Read/write rules file** – specifies how FrameMaker translates from markup (XML files) to the authoring environment and back.

There are many ways to develop and maintain structure applications. DITA-FMx is set up to work with three types of apps, a Topic app and a Map app (both used for authoring) and a Book app (used for printing and PDF generation). If you want to control the way your DITA files are printed, you will want to modify the Book app. If you want to modify the structure or add/remove elements, you'll need to modify both the Topic and Book apps (and possibly the Map app). The default DITA-FMx application folder structure puts the files for all three of these apps into Topic, Map, and Book folders under a main folder named "DITA-FMX_1.2." We find this folder structure to be the most efficient, but you are free to devise any structure that works for your needs.

When creating your own custom structure applications, you can either start by cloning the existing default DITA-FMx apps or by creating your own from scratch. The cloning method is the easiest and least likely to have errors, but you can do which ever works best for you.

IMPORTANT: *You may be tempted to make minor modifications to the default DITA-FMx apps. We strongly recommend that you maintain an instance of the DITA-FMx structure applications in their original form. Because these applications work properly with DITA-FMx, as you customize/build your own apps, you may need to refer back to the originals in case something goes wrong (which it often does).*

RELATED INFORMATION:

"DITA-FMx Specific Element Definitions" on page 29

Cloning the Default Structure Applications

The first step in customizing the layout or formatting of a structure application is to create your own application; the easiest way to create your own is to clone the existing apps.

PREREQUISITES:

First, copy the *FrameMaker\Structure\xml\DITA-FMx_1.2* folder (with all subfolders included) to a new folder name, for our purposes here, we'll call it

MyCo-DITA-1.2. In the new MyCo-DITA-1.2 folder are the three application folders (Topic, Map, and Book), and the DTD folder.

You'll be modifying files in the application folders, and leaving the DTD folder as it is (unless you're specializing, which we won't go into here). You'll want to make the same initial changes to each of the applications (replace *APP* with the Topic, Map, or Book application in the steps below).

Task

1. In the new *MyCo-DITA-1.2\Topic* folder open the *topic_1.2.edd.fm* file in FrameMaker. At the top of the file, you'll see a line that says "Structured Application: DITA-FMx-Topic-1.2", change that to "MyCo-DITA-1.2-Topic". Save this file, but keep it open for now.
2. In the *MyCo-DITA-1.2\Topic* folder open the *topic_1.2.template.fm* file in FrameMaker. Choose **File > Import > Element Definitions**, select the *topic_1.2.edd.fm* EDD file from the list and choose Import.
3. Save and close the template, then save and close the EDD.
4. In the *MyCo-DITA-1.2\Topic* folder open the *struct-apps-stub_topic_1.2.fm* file in FrameMaker. Change all instances of "DITA-FMx" to "MyCo-DITA".

   Additional Information: This involves modifying the text of a variable. You'll actually want to create a new variable and use that, don't just modify the content of the existing variable; if you modify the content of the variable, it may break other applications since this "stub" file is imported into the structure application definitions file.

   a) Open the Variable dialog (**Special** > **Variable**).
   b) Select "DITA-FMx-Path" from the list and choose Edit Definition.
   c) In the Edit user variable dialog, change the name from "DITA-FMx-Path" to "MyCo-DITA-Path", then in the Definition field change "DITA-FMx" to "MyCo-DITA". Then choose Add then Done. Choose Done again in the Variables dialog.
   d) Double-click the first DITA-FMx-Path variable and in the Variables dialog select "MyCo-DITA-Path" and choose Replace.
   e) Now copy and paste this new variable overall instances of it in the stub file (in the Template, DTD, Read/Write Rules, and numerous places in Entity Locations).
5. Save and close the stub file.
6. Now open the structure application definitions file with the **Structure Tools > Edit Application Definitions** command.
7. Place the insertion point just after the Version element (use the Structure View window to see this location easily), and choose **File > Import > File**. Navigate to the new *structapps-stub_topic_1.2.fm* file and select

it. In the Import Text Flow By Reference dialog, accept the defaults and choose Import.

AFTER COMPLETING THIS TASK:

Repeat this task for each of the applications, Topic, Map, and Book. Once you have updated each application, you'll have created a "clone" of the default applications and are now ready to customize them as needed.

Customizing the Formatting of the Default Structure Applications

The formatting provided by the default DITA-FMx structure applications is sufficient for producing nice looking PDFs, but it's common that each user will want to customize the formatting to adhere to house styles. Where to start and what to do can be a challenge.

DITA-FMx uses three structure application types, Topic, Map, and Book. The Topic application is used for authoring topics, the Map application is used for authoring maps, and the Book application is used for publishing.

Assuming that you need to customize the appearance of the published content, you'll need to modify the Book application. If your changes are minor (fonts, headers, footers, etc.), you may be able to just modify the Book template, but if you need to make more substantive changes, you'll need to modify the EDD as well. Remember that if you do make changes to the EDD, you must import the modified EDD into the template.

One of the core concepts of DITA (XML in general) is that you're authoring content that may be published to various types of output formats. Because of this it doesn't typically make sense to author in a WYSIWYG view, and if you were authoring in a view that resembled one of the output formats, you may spend time crafting the content for that output type rather than authoring in an output-agnostic mind-set.

If it is important that your authoring view matches that of the "print" (PDF) output, you can set up the Topic template to mirror the formatting in your Book template, but typically, you can just use the default Topic application in its default state.

There's typically no need to make any modifications to the Map application since it (like the Topic app) is just used for authoring, and has no effect on the appearance of the published documents.

The one time that you will need to modify the Topic and Map applications is if you're creating or modifying the data model itself by either adding or removing elements. If you do decide to modify the data model, be sure to do that in a way that doesn't "break" DITA. While a modified model may work fine for your needs, if you've done it in a way that's not valid (for DITA), your files may not work in other tools that support DITA (because you no longer have DITA-compliant files).

Working with the EDD files

In the default DITA-FMx structure applications, the Topic and Book EDDs share common element definitions using text insets (in the *DITA-FMx_1.2\common* folder). While this does make it a bit more complicated to make edits, it ensures that the two models remain in sync.

The text inset source files are named based on the DITA domain for the elements. If you're not sure where an element definition can be found, you have two options:

- Open the Topic or Book EDD, scroll to locate the element definition (can't search on text of an inset), then double-click the inset and choose "Open" in the Text Inset Properties dialog.
- Open the *dita_1.2.book* file in the *common* folder, then use the Find command to search for text in the book components. This book file is just provided as a tool to locating content in the insets.

After modifying an element definition in the inset source file, save the file and return to the main Topic EDD, then scroll to the inset location in the EDD and double-click the inset. In the Text Inset Properties dialog choose "Update Now." Updating text insets in a large EDD can take some time, so don't be alarmed when you have to wait for a bit (possibly a minute or two) before the inset has been updated.

REMEMBER: *Always update both the Topic and Book EDDs when changing any of the shared inset source files.*

Some of the element definitions in the shared insets have conditional tagging applied which hides/shows certain context rules or general rules in the respective Topic or Book EDDs. The shared files have all conditions visible, and the Topic and Book EDDs have the hide/show properties set properly. If you change the hide/show settings in the EDDs, it may take considerable time for the change to take effect (more than 5 minutes is not unexpected). It's best to leave the conditional settings alone in these files.

You may also run into the use of variables within an EDD. This is often used to change the text of a general rule for use in different EDDs. The variable definition in the "container" EDD will change the value of the variable in the inset EDD. Pay careful attention to this when making edits.

IMPORTANT: *After making any modifications to an EDD, its element definitions must be imported into the associated template file. The structure application definition references the template, not the EDD, so that's the only way your changes will be available.*

Where to apply formatting, the EDD or the template?

The template file defines the named styles and formats, as well as the overall page layout for a component (FM chapter file). The EDD declares element definitions, which includes the element type, general rule, attribute definitions, and context rules.

The context rules allow you to specify the formatting that is applied to the element under different situations. These situations (contexts) may be based on things like the "level" of the element (how deeply nested it is), the type of ancestor elements, the position within the parent element, or attributes assigned to the current or ancestor elements. When a context is matched the rule can assign a named style (defined in the template) or it can directly set any property available through the FrameMaker user interface (well, almost any property), or both.

This means that you can apply formatting through named styles in the template (based on context rules in the EDD), or by directly assigning the properties in the EDD. In most cases (as is the case in the default DITA-FMx apps) you'll do some of both. While this isn't really the best situation for ease of maintenance, you'll find that it's just easier to implement.

This "mixed model" can be the cause of some confusion when trying to modify an existing structure application. You'll want to change the text that displays when you insert an element (like the text "Task" when you insert a <steps> element in a task). In order to determine where that text is defined you have to do some digging. It may be part of an Autonumber Format in the paragraph style (in the template or EDD), it may be defined as a prefix (in the EDD), or it may be included in a reference frame (defined in the template, but possibly assigned in the EDD).

This is just part of the "joy" that is structure application development. It's not hard once you understand what's going on, but it can take some time to get to that point, especially when working on a structure application developed by someone else.

Don't forget that you can ask questions from the "community"! If you can't figure out where something is defined, email the *dita-fmx-users* Yahoo group for help.

Understanding the Book application model

While you can make use of multiple "Topic" structure applications for authoring (if using the "doctype/application mapping" feature), all publishing for a single deliverable must be done using a single Book structure application. Because it's possible that you're combining multiple topic types into a single FM chapter file, the Book application needs to be designed to support all of those topic types in a single model.

While this may sound complicated, it's simplified by the fact that this "Book" model doesn't need to strictly adhere to the DITA model. It is intended as an output format, and not to be round-tripped back to DITA XML. In the extreme sense, you could make a Book EDD where all of the general rules are "<ANY>" (meaning that anything is valid). Remember that because this is an output model, you don't need to worry about validation. You just need to determine the complete set of topic models that will be used in your book, and add all of those element definitions to the EDD.

The *fmx-book_1.2.dtd* (the DTD associated with the Book application) is created by saving the Book EDD to a DTD (**Structure Tools > Save as DTD**).

Adding Map to Book Metadata Mappings

Extending the default metadata mappings allows you to pass custom element and attribute data from the map to the generated book and component files.

Two DITA-FMx features read the attribute values on the <fm-ditabook> element (the root of a generated book file). These attributes can define values that are passed to the book components as FrameMaker variables. These attributes can also be used to control the show/hide state of conditions in the generated book components. For information on enabling these features see the ImportAttrsAsVars, and ImportAttrsAsConds parameter information in Book-Build INI file.

In order to add your own metadata mappings, you need to modify a number of files in the Book structure application. The following is an overview of the steps required; details are provided below.

1) Modify the Book EDD to add new attributes to the <fm-ditabook> element definition, then import the updated EDD into the Book template.
2) Modify the *fmx-book_1.2.dtd* file to add the new attribute definitions.
3) Modify the *bookmap2fmbook.xsl* file to copy the desired metadata (attributes or element content) into the new <fm-ditabook> attributes.

Specific information regarding the modification of the DTD and XSL files are provided below.

Modifying the fmx-book_1.2.dtd file

The *fmx-book_1.2.dtd* file is found in the *dtd-fmx* folder.

1) Open the *fmx-book_1.2.dtd* file and locate the attribute definitions for the <fm-ditabook> element. The unaltered attribute definition should look like the following.

```
<!ATTLIST fm-ditabook
    href CDATA #IMPLIED
    format CDATA #IMPLIED
    title CDATA #IMPLIED
>
```

2) Add the new attributes. The example below adds a new “version” attribute.

```
<!ATTLIST fm-ditabook
    href CDATA #IMPLIED
    format CDATA #IMPLIED
    title CDATA #IMPLIED
    version CDATA #IMPLIED
>
```

3) Save and close the DTD file.

Modifying the bookmap2fmbook.xsl file

The *bookmap2fmbook.xsl* file is used to aggregate the DITA topics into a book file and component FM files. It can be used to copy data from the DITA map into attributes of the fm-ditabook element. Most likely you would copy data from the bookmeta or topicmeta elements, but any accessible element or attribute data could be used.

Open the *bookmap2fmbook.xsl* file in a text editor. The default import XSLT file contains some default mappings that can be copied and modified to create your own. To locate this code, scan the XSLT file for the following:

```
<xsl:attribute name="created">
  <xsl:value-of select="translate(//critdates/created[1]/@date,'&#10;',' ')"/>
</xsl:attribute>
```

Just copy the entire xsl:attribute block (3 lines), change the @name attribute to the value of the fm-ditabook attribute you want to create, then change the xsl:value-of/@select code to grab the value of the attribute or element you want to be assigned to the new attribute. You’ll see examples of both element and attribute matching, plus code that locates the first or last instance of that value (in case there may be multiple).

You’ll need to add a similar attribute definition for each type of metadata that you want to extract from the map. Note that even if you don’t expect there to be more than one instance of a metadata value, it’s a good practice to explicitly select the first (or last) value, just in case there are more than one in a file you’re processing.

Also note that these attribute definitions are found in two areas in the XSLT file, in the map template and the bookmap template. You’ll want to modify both or either as needed.

RELATED INFORMATION:

"Passing Map-level Metadata to the FM Book" on page 68

DITA-FMx Specific Element Definitions

To make the most of the authoring and publishing features in DITA-FMx, some elements must be set up in a specific manner. This topic describes those requirements.

The easiest way to get up and running with DITA in FrameMaker is to use the default structure applications provided with DITA-FMx; they will "just work" right out of the box. Once you're comfortable with things and want to make some adjustments to fit your house style or want to specialize, you can either clone the apps provided, use some DITA apps provided by someone else, or make your own from scratch.

If you want to customize the apps provided with DITA-FMx, refer to the topic Developing Custom Structure Applications. If you want to use other applications or want to make your own, just be sure that they adhere to the requirements expected by DITA-FMx. In order to implement certain DITA features in DITA-FMx, the applications must be set up in a very specific way.

The information in the following topics describes the elements and other requirements that need to exist in a DITA structure application so that it functions properly with DITA-FMx. All of the modifications described affect the structure application files only (EDD, template, rules file), no modifications need to be made to the DITA DTD files. The items are grouped by feature, some of these features require the addition of multiple elements or structure application modifications. If you don't want to make use of a given feature, you can skip the modifications described in that section. There are many other rules and EDD modifications that are needed for proper use of DITA files in FrameMaker, these are not discussed; only those features that are required or used by DITA-FMx are documented at this time.

The easiest way to set up these features in your custom application is to copy and paste from the default DITA-FMx applications.

RELATED INFORMATION:

"Developing Custom Structure Applications" on page 21

fm-reflabel and Topic References

In order to provide a clickable label for some of the topic referencing elements in map and bookmap files, the <fm-reflabel> element must be added to the EDD. This element is discarded on file save, and is recreated on file open.

To implement this feature the <fm-reflabel> element definition must be added to the EDD and added to the general rule of the <topicref>, <chapter>, <appendix>, <part>, etc. elements. The <fm-reflabel> element should have a general rule of "<TEXT>". If you want to apply character formatting you can add text format rules.

Because the <fm-reflabel> element is not part of the DITA specification, it must be deleted on file save. Add the following rule to the Map application rules file:

```
fm element "fm-reflabel" drop;
```

If this element is not added to the EDD, you will be able to create topic referencing elements, but they will have no label to click on.

For maps based on the DITA 1.2 model, most of the topic referencing elements will use the <navtitle> element instead of the <fm-reflabel> element, however, the <keydef> element does need a clickable label so uses <fm-reflabel>.

The <fm-reflabel> element should be added only to Map application.

NOTE: *The <fm-reflabel> element replaces the <fm-topicreflabel> as of DITA-FMx 1.1. This change was made to support a clickable label on all topic referencing bookmap elements. When migrating FM-DITA applications, you'll want to change <fm-topicreflabel> to <fm-reflabel>.*

reltable

Because DITA defines the <reltable> element with the same structure as a <simpletable> element, it requires additional internal structure to be rendered properly as a table within FrameMaker. This adds the <fm-reltablehead> and <fm-reltablebody> elements. Also, in order to support the use of the <topicmeta> element as the child of a <reltable>, the <fm-reltablemeta> element is added. On file save, the <fm-reltablemeta> element is converted into a <topicmeta>, and on import a <topicmeta> that is the child of a <reltable> is converted into an <fm-reltablemeta> element.

In the EDD, general rule for <reltable> should be "(fm-reltablemeta)?, (fm-reltablehead)?, (fm-reltablebody)". The <fm-reltablehead> elementis defined as a Table Heading element named and the <fm-reltablebody> element is defined as a Table Body element. The <fm-reltablehead> element specifies a general rule of "relheader" and the <fm-reltablebody> element specifies a general rule of "relrow+". The <fm-reltablemeta> element is defined as a Table Title element and has the same general rule as the <topicmeta> element. It also has the same attribute definitions as the <topicmeta> element, but no other element definition properties are needed for <fm-reltablemeta>.

In addition to the EDD modifications, the following rules are needed for the <reltable> structure:

```
fm element "fm-reltablehead" unwrap;
fm element "fm-reltablebody" unwrap;

fm element "fm-reflabel" drop;

element "fm-reltablemeta"
{
  is fm table title element;
}

element "reltable"
{
  is fm table element;
}
element "relheader"
{
  is fm table row element;
  fm property row type value is "Heading";
}
element "relrow"
{
  is fm table row element;
  fm property row type value is "Body";
}
```

The initial rules (first two lines) discard the head and body wrappers on file save. The next line discards the <fm-topicreflabel> element. The remaining rules define <fm-reltablemeta> as a Table Title, the root table element (<reltable>), the row element that is a "Heading" row (<relheader>) and the row element that is a "Body" row (<relrow>).

fm-xref

The <fm-xref> element is used to create a FrameMaker-style cross-reference used inline as a standard DITA <xref> element would be used. On file save, this converts to a standard DITA <xref> element and on file open, it converts back into an <fm-xref> element.

The way an <fm-xref> is identified as such is the type attribute value which is given the format of "fm:*FORMAT-NAME*". Where *FORMAT-NAME* is the cross-reference format name as defined in the FrameMaker template. When an <fm-xref> element is inserted, the standard FrameMaker Cross-Ref dialog displays, select the target element and the cross-ref is inserted as the <fm-xref> element.

To use the <fm-xref> feature requires the creation of the <fm-xref> element definition.

1) Create an <fm-xref> element definition (in the EDD):
   a) Make a copy of the default <xref> element definition.
   b) Change the name to "fm-xref"
   c) Delete the general rule (an <fm-xref> cannot have child elements)
2) Add <fm-xref> to the appropriate general rules (everywhere that an <xref> element is valid, the <fm-xref> element should also be).

The <fm-xref> element should be added to both the Topic and Book applications.

fm-link and fm-linkref

The <fm-link> element is used to create a FrameMaker-style cross-reference as a related-links item. On file save, this converts to a standard DITA <link> element and on file open, it converts back into an <fm-link> element.

The way an <fm-link> is identified as such is the type attribute value which is given the format of "fm:*FORMAT-NAME*". Where *FORMAT-NAME* is the cross-reference format name as defined in the FrameMaker template. When an <fm-link> element is inserted, the standard FrameMaker Cross-Ref dialog displays, select the target element and the cross-ref is inserted as a <fm-linkref> element (child of <fm-link>). You can optionally add a standard <desc> element as a sibling of the <fm-linkref>.

To use the <fm-link> feature requires the creation of <fm-link> and <fm-linkref> elements.

1) Create an <fm-link> element definition (in the EDD):
   a) Make a copy of the default <link> element definition.
   b) Change the name to "fm-link"
   c) Change the general rule to "fm-linkref, desc?"
   d) Change the default value for the class attribute to "- topic/fm-link"
   e) Add an automatic insertion of a child "fm-linkref" element.
2) Create an <fm-linkref> element definition (in the EDD):
   a) Make a copy of the default <link> element definition.
   b) Change the name to "fm-linkref"
   c) Change the element type from Container to CrossReference
   d) Delete the general rule
   e) Make all of the attributes into the type "String" that is "Optional" with a Special Attribute Control of "Hidden". Remove all default values.
3) Add <fm-link> to the appropriate general rules (everywhere that a <link> element is valid, the <fm-link> element should also be).

The <fm-link> and <fm-linkref> elements should be added to both the Topic and Book applications.

link and fm-linklabel

In order to provide a clickable label for the <link> element, the <fm-linklabel> element must be added to the EDD. This element is discarded on file save, and is recreated on file open.

The only thing required to implement this feature is to add the element definition to the EDD and to add it to the general rule of the <link> element. The <fm-linklabel> element should have a general rule of "<TEXT>". If you want to apply character formatting you can add text format rules.

If this element is not added to the EDD, you will be able to create <link> elements, but they will have no label to click on.

NOTE: It has been found that children of this element may not import properly if your EDD specifies a prefix rule for the <link> element. The FM8-DITA and early DITA-FMx Topic applications specify a prefix rule for <link>. If you're having link import problems, you may want to investigate this possibility.

The <fm-linklabel> element should be added to both the Topic and Book applications.

indexterm and fm-indexterm

The <fm-indexterm> element represents an <indexterm> element as a FrameMaker marker. If the **indexterm to fm-indexterm** option is enabled in the Options dialog, <indexterm> elements and any index-related child elements are converted into the FrameMaker index marker syntax and added to an <fm-indexterm> element.

In order to support the <indexterm> to <fm-indexterm> conversion, the Topic and Book structure applications must have the <fm-indexterm> element defined. This is a clone of the <indexterm> element, but is defined as a Marker object rather than a Container. All general rules that allow the <indexterm> element should also allow the <fm-indexterm> element. The default value of the <fm-indexterm> element's class attribute should be "- topic/indexterm fmx/fm-indexterm". The <fm-indexterm> marker only supports the conversion of <indexterm> elements that have the following child elements: <index-term>, <index-see>, <index-see-also>, and <index-sort-as>. If your DITA files use other child elements you should disable this optoin.

No rules are needed to properly process the <indexterm> or <fm-indexterm> elements. In fact, if you're migrating an older structure application you should remove the <indexterm> rule, it is no longer needed.

If the **indexterm to fm-indexterm** option is disabled, <indexterm> elements will open as simple container elements.

fm-data-marker

The <fm-data-marker> element round-trips to DITA as the <data> element, and allows you to make use of multiple marker types in FrameMaker. When you insert an <fm-data-marker>, that element saves to DITA as a <data> element with the @datatype attribute set to "fm:marker" and the @name attribute set to the marker's type. The marker text is assigned to the @value attribute.

The element definition of <fm-data-marker> is identical to that of the the <data> element, except that the object type is a Marker instead of a Container. Also the @class attribute is "topic/data fmx/fm-data-marker." The plugin handles the conversion from <fm-data-marker> to <data> and back; no read/write rules are needed for this element.

fm-var

The <fm-var> element round-trips to DITA as a <ph> element and allows you to make use of FrameMaker variables in DITA. When you insert a FrameMaker variable it is wrapped in a <fm-var> element on file save. That element saves to DITA as a <ph> element with the @outputclass attribute set to "fmvar:*VARNAME*". The content of the <fm-var> element persists in the <ph> element so it can be processed outside of FrameMaker.

The element definition of <fm-var> is identical to that of the the <ph> element. DITA-FMx handles the conversion from <fm-var> to <ph> and back; no read/write rules are needed for this element.

image

To support the image/alt element the value of the <alt> element is stored in the anchored frame's "object attribute" while open in FrameMaker. Select the anchored frame, then choose **Graphics > Object Properties**, then Object Attributes. The <alt> value is stored in the "Alternate" field. The EDD should define the <image> element as a "Graphic," which doesn't allow for a general rule (hence no support for the <alt> and <longdescref> child elements in FrameMaker).

To support the proper sizing and placement of <image> elements, add the following to the "image" rule in both Topic and Book apps:

```
element "image"
{
  is fm graphic element;
  fm property import by reference or copy value is "ref";

  writer facet default
  {
    specify size in pt;
  }
  attribute "href"
  {
     is fm attribute "href";
     is fm property file;
  }
  attribute "align"
  {
    is fm property alignment;
    value "left" is fm property value align left;
    value "right" is fm property value align right;
    value "center" is fm property value align center;
    value "current" is fm property value align center;
  }
  attribute "height"
  {
    is fm property height;
    is fm attribute;
  }
  attribute "width"
  {
    is fm property width;
    is fm attribute;
  }
}
```

simpletable-Based Tables

Tables based on the <simpletable> element require a slightly different structure than that provided by DITA in order to render properly as a table in FrameMaker. To accommodate this difference, these tables need an extra level of hierarchy added to define the table heading area and the table body area while working in FrameMaker.

The <simpletable> element and those elements that are specialized from <simpletable> (<properties> and <choicetable>) need a Table Heading element named "fm-*TABLETYPE*head" and a Table Body element named "fm-*TABLETYPE*body" added to the EDD. The <fm-simpletablehead> element specifies a general rule of "sthead" and the <fm-simpletablebody> element specifies a general rule of "strow+". The general rule for each of the *TABLETYPE* elements should be "fm-*TABLETYPE*head?, fm-*TABLETYPE*body".

In addition to the EDD modifications, the following rules are needed for each <simpletable>-based structure:

```
fm element "fm-simpletablehead" unwrap;
fm element "fm-simpletablebody" unwrap;

element "simpletable"
{
  is fm table element;
  attribute "relcolwidth" is fm property column widths;
}
element "sthead"
{
  is fm table row element;
  fm property row type value is "Heading";
}
element "strow"
{
  is fm table row element;
  fm property row type value is "Body";
}
```

The initial rules (first two lines) discard the head and body wrappers on file save. The remaining rules define the root table element (in this case <simpletable>), the row element that is a "Heading" row (in this case <sthead>) and the row element that is a "Body" row (in this case <strow>). For <simpletable> specializations, you must provide these three rule groups so the row elements are placed into the right "fm-*TABLETYPE*head" or "fm-*TABLETYPE*body" elements.

Because the <choicetable> element must have only 2 columns, the root table rule looks like the following:

```
element "choicetable"
{
  is fm table element;
  fm property columns value is "2";
  attribute "relcolwidth" is fm property column widths;
}
```

And because a <properties> table can have one, two, or three columns, the <properties> element read/write rule is no longer used (this old rule required that a <properties> table always have three columns). The Element Mapping dialog in Options allows the definition of multiple optional table cell elements, and can be used to support other specializations of <simpletable>.

The <fm-simpletablehead>, <fm-simpletablebody>, and the head/body elements for other <simpletable>-specialized elements should be added to both the Topic and Book applications.

RELATED INFORMATION:

"Element Mapping" on page 55

DITA-FMx Specific Book Application Elements

The Book structure application includes a number of special (non-DITA) elements that are needed for proper rendering of certain features.

The Book application is essentially identical to the Topic application except that it has additional elements that are used to isolate the separate pieces that are part of the book. The Book application also makes use of special <fm-*> elements for proper rendering of various features. Without these <fm-*> elements, it's likely that your book build will fail.

Generated book elements

These elements are used in the structure of the generated book file. All are required.

fm-ditabook

The root element of the generated book file. Also represents the root DITA map used to create the book. The @href attribute defines the path and name of the source ditamap file. Additional attributes may be added from map metadata for use as variables or by element definition context rules.

fm-bookcomponent

Used as a container for generated book components. This wraps any generated files that are added by the book-build process, or you can add this for any generated files you add manually.

Generated file (chapter) elements

These elements are used to support various features in generated file or chapter components. Only the <fm-ditafile> element is required; the rest are used if your source topics include elements that make use of those features.

fm-ditafile

Identifies each DITA topic file. The @href attribute defines the path and name of the source topic file. The @maptype attribute contains the name of the element that referenced this topic file in the original ditamap. This is used by the book-build process to identify specific book components to automatically apply templates as well as numbering and pagination properties.

fm-tabletitle

Allows for properly formatted table titles in generated FrameMaker files. The Book application must have the <fm-tabletitle> element defined and valid as a child of the <tgroup> element.

fm-figuretitle

Supports <fm-xref> references to "moved" figure titles in generated FrameMaker files. The Book application must have the <fm-figuretitle> element defined and valid as the last child of the <fig> element. If this element is not defined, the "move figure titles" book-build option will use the default <title> element, but in order to support cross-references to figure titles that are moved, this element must be defined and have its @id attribute's type set to "UniqueID."

fm-figuredesc

Supports content in a <desc> element that is "moved" with figure titles in generated FrameMaker files. The Book application must have the <fm-figuredesc> element defined and valid as the last child of the <fig> element. If this element is not defined, the "move figure titles" book-build option will create this element.

fm-hazardtable

If "hazard to table" processing is enabled in the book-build INI file (the default), the <hazardstatement> element model is wrapped in a special hazard table structure. The <fm-hazardtable> element is the container for this hazard table.

Compare to the <table> element. The attributes from the <hazardstatement> element are copied to this element and the <hazardstatement> element is deleted. The general rules for all elements that contain the <hazardstatement> element must be updated to also support the <fm-hazardtable> element.

fm-hazardtgroup

Compare to the <tgroup> element.

The default element definition in the Book EDD inserts a table format named "Hazard." If you want all of your hazard tables to use the same style, you can leave this table format as it is. But if you'd like different styles for each hazard statement type, you can modify the EDD to insert a different table format based on the fm-hazardtable/@type attribute.

fm-hazardthead

Compare to the <thead> element. (Defined in the Book EDD, but not currently used.)

fm-hazardtbody

Compare to the <tbody> element.

fm-hazardrow

Compare to the <row> element.

fm-hazardentry

Compare to the <entry> element.

fm-hazardhead

Provides a heading label for the hazard table.

fm-hazardsymbolwrapper

Wraps the <hazardsymbol> element that is inserted into the first column of the hazard table. Also, inserted into the first column even if there is no <hazardsymbol> element (can be used to control formatting or insertion of a reference image).

Modified elements

Other than the special <fm-*> elements that are unique to the Book application, the following element definitions vary from those in the Topic application:

title

The context rule level numbering differs to allow for an additional level of "chapter titles."

fig

Includes the <title> and <fm-figuretitle> elements as optional elements at the end of the general rule. This is required by the "Move fig/title to end of fig element" Book Build option.

tgroup

Includes the <fm-tabletitle> element in the general rule. This is required by the "Move table/title to table/tgroup/title" Book Build option.

typeofhazard

Context rule checks the @type attribute of the <fm-hazardtable> element instead of the <hazardstatement> element.

Publishing with the DITA Open Toolkit

Details on setting up DITA-FMx to communicate with the DITA Open Toolkit.

The **Generate Output** command lets you publish output through the DITA Open Toolkit. This command provides two main options: generate output from the current topic or map, or by using a specific Ant target. The setup required for these two options is described below.

DITA-FMx makes a call to the DITA Open Toolkit (DITA-OT) through a batch file named *~ant-build.cmd*. This file is generated in the user's DITA-FMx folder (use the **Open DITA-FMx Folder** command). In order for this batch file to work properly, the environment variables that define the location of Java and Ant (and possibly other utilities) must be properly defined.

The **DITA-OT Directory** and the **DITA-OT Environment Setup File** fields in the External Application Settings dialog must be specified to load the proper environment settings when the DITA-OT is run. Setup varies depending on the version of the DITA-OT. OT 1.x requires some additional setup, but OT 2.x and 3.x are quite simple.

Complete the following steps to enable publishing through the DITA-OT:

1) Download the latest ZIP of the DITA-OT <www.dita-ot.org/download>. Extract the contents of this file to your local file system into a directory named *C:\DITA* (or similar). It is a good practice to ensure that the path to the DITA-OT directory has no spaces in any of the directory names.

After extracting the contents of the OT archive, you will have a path such as *C:\DITA\dita-ot-3.5.4*.

2) Make sure you've got a recent version of Java Development Kit. We recommend using OpenJDK from adoptopenjdk.net, but other compatible options are listed in the DITA-OT documentation (www.dita-ot.org).

3) Your JAVA_HOME environment variable should be set to the path of the JDK (perhaps "C:\Program Files\Java\jdk1.8.0_65"). This will typically be handled by the JDK installation program.

4) *DITA-OT 1.x only:*

   a) Copy the *ditafmx-ant.xml* file from the *Program Files\Adobe\FrameMaker\DITA-FMx-2* folder to your DITA-OT folder.

   b) In the DITA-OT folder, copy the *startcmd.bat* file to *ditafmx-startcmd.bat* (in the same folder copy the file to a new name).

   c) Open the *ditafmx-startcmd.bat* file in Notepad (or other text editor) and comment out the last line (add a "REM" at the start of the line). If you don't comment out this line, it will still work, you'll just get an empty shell that hangs around each time you build. It should look like this:

```
REM start "DITA-OT" cmd.exe
```

5) In the DITA Options dialog, select the **External Apps** button to display the External Application Settings dialog. Set the value of the **DITA-OT Directory** option to reference the main DITA-OT directory. Then set the value of the **DITA-OT Environment Setup File** option to reference the *bin\dita.bat* file (for OT 2.x or later) or the *ditafmx-startcmd.bat* file (for OT 1.x).

   NOTE: *For DITA-OT versions prior to 1.6 as well as versions 3.1.2 and 3.1.3, you'll be prompted to enter the OT version. These versions do not expose a way for DITA-FMx to know the version number. This is required.*

   (*OT 1.x only*) If you make use of the **Selected Target** option with the **Generate Output** command, this environment setup file is used for those builds as well. If you want to specify an alternate environment setup file, you can add the "EnvironmentSetup" parameter to the associated "ANT:<targetname>" sections.

If you're using DITA-OT 3.2 or later, the currently installed plugin targets will automatically be listed in the Current File list in the Generate Output dialog. This is done when the DynamicTargets option is enabled in the BuildFile section of the *ditafmx.ini* file (the default). If you're using an older version of the

DITA-OT or have this option disabled, you'll need to make sure the list of targets in the BuildFile section includes the targets you plan to use.

(*OT 1.x only*) You may need to start Frame from a shell in the OT directory, or add a "cd %DITA_DIR%" line to the *ditafmx-startcmd.bat* file. Alternatively, you may want to start FM from the DITA-OT directory. To do this just create a shortcut to *FrameMaker.exe*, and in the "Start In" field of the Shortcut tab, set this value to the path to your DITA-OT directory.

Generate Output: Current File Option (OT 2.x or later)

The **Current File** option of the **Generate Output** command lets you generate output from the current topic or map. The information about the current file and selected target are passed to the build process through command line parameters.

Most of the information below is only useful if you're not using the DynamicTargets option (described earlier). This provides for additional control over the publishing targets available in the Generate Output dialog.

By default DITA-FMx provides support for 8 publishing targets. To add or modify this list, edit the BuildFile section of the *ditafmx.ini* file. The following is the default BuildFile section after completing the basic setup (assuming OT 2.5.4):

```
[BuildFile]
EnvironmentSetup=C:\DITA\dita-ot-2.5.4\bin\dita.bat
DitaDir=C:\DITA\dita-ot-2.5.4\
AntCommand=ant
AntScript=ditafmx-ant.xml
Count=8
1=xhtml
2=html5
3=pdf
4=htmlhelp
5=javahelp
6=eclipsehelp
7=tocjs
8=troff
```

NOTE: *The AntCommand and AntScript parameters are ignored when using the DITA-OT 2.x or later.*

Each of the numbered entries corresponds to target (transformation type) defined in the OT. If you want to add additional targets, install the necessary OT plugin to support those targets, then add the corresponding target name to the BuildFile section (don't forget to update the Count parameter).

As of DITA-FMx 2.0.07 three additional BuildFile parameters are available:

OT2Params

Pass additional parameters to the OT transformation.

NOTE: *Specifying an output location (via '-output=' or '-o='), will override the creation of the default build directory based on the current filename.*

OT2Params-<build>

Same as the OT2Params parameter, but only applies to the specified DITA-FMx build. If set, this value overrides the OT2Params value. For example, the parameter name could be: OT2Params-xhtml

OT2Logging

Possible values are: Verbose (the default value, if not specified), Debug, or None (errors only).

If you need to use a different DITA-OT installation for a specific build target, add new entries for EnvironmentSetup-<target> and DitaDir-<target>. For example, if the “chm” target required the use of a specially set up DITA-OT installation, you might use the following:

```
[BuildFile]
EnvironmentSetup=C:\DITA\dita-ot-2.5.4\bin\dita.bat
DitaDir=C:\DITA\dita-ot-2.5.4\
EnvironmentSetup-chm=C:\DITA\dita-ot-2.1.1-special\bin\dita.bat
DitaDir-chm=C:\DITA\dita-ot-2.1.1-special\
AntCommand=ant
AntScript=ditafmx-ant.xml
Count=9
1=xhtml
2=html5
3=pdf
4=chm
5=htmlhelp
6=javahelp
7=eclipsehelp
8=tocjs
9=troff
```

In order to take advantage of ditaval filtering, you must register or create ditaval files so they are known to DITA-FMx. The Ditaval Manager provides the means to add existing files, or create new files.

Generate Output: Current File Option (OT 1.x)

The **Current File** option of the **Generate Output** command lets you generate output from the current topic or map. The information about the current file and project are passed to the Ant script through command line parameters.

By default DITA-FMx provides support for 8 publishing targets. To add or modify this list, edit the BuildFile section of the *ditafmx.ini* file and modify the entries in the *ditafmx-ant.xml* file. The following is the default BuildFile section after completing the basic setup:

```
[BuildFile]
EnvironmentSetup=C:\DITA\DITA-OT1.8.5\ditafmx-startcmd.bat
DitaDir=C:\DITA\DITA-OT1.8.5\
AntCommand=ant
AntScript=ditafmx-ant.xml
Count=8
1=xhtml
2=html5
3=pdf
4=htmlhelp
5=javahelp
6=eclipsehelp
7=tocjs
8=troff
```

Each of the numbered entries corresponds to a target defined in the *ditafmx-ant.xml* file. If you want to add additional targets, add it to the *ditafmx-ant.xml* file, then add the corresponding target name to the BuildFile section (don't forget to update the Count parameter).

NOTE: *If you're using a version of the DITA-OT earlier than 1.6, you'll be prompted to enter the version number during OT setup (in the DITA Options dialog). This assigns that value to the BuildFile/OTVer key in the ditafmx.ini file.*

If you need to use a different DITA-OT installation for a specific build target, add new entries for EnvironmentSetup-<target> and DitaDir-<target>. For example, if the "chm" target required the use of a specially set up DITA-OT installation, you might use the following:

```
[BuildFile]
AntCommand=ant
EnvironmentSetup=C:\DITA\DITA-OT1.8.5\ditafmx-startcmd.bat
DitaDir=C:\DITA\DITA-OT1.8.5\
EnvironmentSetup-chm=C:\DITA\DITA-OT1.8.5-MYCHM\ditafmx-startcm
d.bat
DitaDir-chm=C:\DITA\DITA-OT1.8.5-MYCHM\
AntScript=ditafmx-ant.xml
Count=9
1=xhtml
2=html5
3=pdf
4=chm
5=htmlhelp
6=javahelp
7=eclipsehelp
8=tocjs
9=troff
```

In order to take advantage of ditaval filtering, you must register or create ditaval files so they are known to DITA-FMx. The Ditaval Manager provides the means to add existing files, or create new files.

Generate Output: Selected Target Option (OT 1.x only)

The following steps should get you up and running with the **Selected Target** option of the **Generate Output** command.

1) Copy the *PROJECT.xml* file to your DITA-OT directory and rename it something appropriate for your project.

2) Edit the *<project>.xml* file and set the properties indicated in the comments so they match your system and project.

3) Set up the *ditafmx.ini* file as follows:

NOTES:

- Be sure that the LogFile parameter specifies a directory that exists before the Ant script is run, otherwise the build will fail.
- The optional EnvironmentSetup parameter can be used to specify a batch file to run before running the Ant script in order to set up the environment. (If this is not added, the default setting in the BuildFile section will be used.)
- In the examples below, swap "<PROJECT>" for the actual project name.
- If the builds aren't working, try running the script from the command line: `ant -f project.xml html`

```
[AntBuild]
Count=3
1=<PROJECT> - CHM
2=<PROJECT> - HTML
3=<PROJECT> - PDF

[ANT:<PROJECT> - CHM]
BuildFile=C:\DITA-OT1.5.4\<PROJECT>.xml
EnvironmentSetup=C:\DITA-OT1.5.4\ditafmx-startcmd.bat
Target=chm
OutputDir=C:\Projects\<PROJECT>\out\chm
Logfile=C:\Projects\<PROJECT>\ant-buildlog-chm.txt

[ANT:<PROJECT> - HTML]
BuildFile=C:\DITA-OT1.5.4\<PROJECT>.xml
EnvironmentSetup=C:\DITA-OT1.5.4\ditafmx-startcmd.bat
Target=html
OutputDir=C:\Projects\<PROJECT>\out\html
Logfile=C:\Projects\<PROJECT>\ant-buildlog-html.txt
```

```
[ANT:<PROJECT> - PDF]
BuildFile=C:\DITA-OT1.5.4\<PROJECT>.xml
EnvironmentSetup=C:\DITA-OT1.5.4\ditafmx-startcmd.bat
Target=pdf
OutputDir=C:\Projects\<PROJECT>\out\pdf
Logfile=C:\Projects\<PROJECT>\ant-buildlog-pdf.txt
```

RELATED INFORMATION:

"Generate Output" on page 78

Setting Up to Use Cross-References

Explains the differences between FrameMaker and DITA based <xref> or <link> elements and how they are used.

DITA-FMx offers two types of cross-references, a DITA-based cross-reference (the xref and link elements) and a FrameMaker-based cross-reference (the <fm-xref> and <fm-link> elements). This distinction only exists within the FrameMaker user interface; when exported to a DITA XML file, both types appear as standard <xref> and <link> elements.

The FrameMaker-based cross-reference is defined as a “Cross-Reference” element in the EDD as opposed to a DITA-based cross-reference which is defined as a “Container” element. FrameMaker-based cross-references take advantage of the formatting capabilities available to standard FrameMaker cross-references such as the inclusion of page numbers and additional text (this formatting is defined in the underlying FrameMaker template). DITA-based cross-references will either display the text of the target element or static “alternate” text that is entered when the cross-reference is created.

FrameMaker-based cross-references can only reference another element, while a DITA-based cross-reference can reference another element or content outside of the local scope such as a website. Both cross-reference types have their purpose and it’s up to the author to decide which suits their needs best.

You can use either or both of these cross-reference types. The provided structure applications are set up to use both types of references.

When saved to PDF, the FrameMaker-based cross-references will become live hyperlinks. In order for DITA-based references to become hyperlinks, you must enable the “Convert Xrefs to Hyperlinks” option in the Book Build Setting dialog. Alternatively, you can run the **Xref to Hyperlink** command after generating the book/FM files.

By default, the structure application is set up to only allow <fm-xref> and <fm-link> elements to reference other topics (such as <topic>, <task>, <concept>, and <reference> elements) and <section> elements. You can customize the EDD to allow referencing of other elements if needed. In order for an element to appear in the FrameMaker Cross-Reference dialog as a cross-reference target, the “id” attribute of that element must be defined as a *type* of “UniqueID.” By default, most id attributes are defined as a type of “String.” Just change the type to UniqueID and that element will be listed in the Cross-Reference dialog.

TIP: *If you are using FM-based cross-references, always use the Source Type of “Elements,” never use a Source Type of “Paragraphs” or “Cross-Ref Markers.” If the elements that you want to reference are not available, you need to update the EDD as described above.*

Advanced Cross-Reference Options

NOTE: *The following information is provided for those who want to change the default functionality for FrameMaker-based cross-references; there is no reason to read further if the default setup is fine for your needs.*

To ensure compliance with the DITA specification, when an <fm-xref> or <fm-link> element is saved to XML, it is saved to the corresponding DITA element. By default the <fm-xref> element is saved to the <xref> element and the <fm-link> element becomes a <link> element. When reopened in FrameMaker, these elements convert back into the proper FM-based element. If needed, you can change the way this process works.

To save to specialized <xref> or <link> element names.

If your DITA model uses a specialized <xref> or <link> elements, as long as the class attribute values indicate the proper inheritance, they should function properly. If you make use of the <fm-xref> or <fm-link> functionality, you must choose only one element for each to convert into on file save. To specify an element other than the default <xref> or <link> for the <fm-xref> or <fm-link> elements make the following changes to the *ditafmx.ini* file:

- Set GeneralExport\XrefProcessing to 1
- Set GeneralExport\CrossRefToXref to 1
- Set GeneralExport\DitaXrefElem to the “xref” element name
- Set GeneralExport\DitaLinkElem to the “link” element name

To write the FM-based cross-reference text to XML.

Normally on file save, the FM-based cross-reference text is not saved to the XML file; it is regenerated on file open. If you want the reference text saved to the XML file, you must modify the *ditafmx.ini* file as follows:

- Set GeneralExport\WriteLinkTextToFmXrefs to 1

To disable all cross-reference processing

If you want to manage all cross-reference processing through read/write rules or XSLT transformations, make the following modifications to the *ditafmx.ini* file:

- Set GeneralExport\XrefProcessing to 0
- Set GeneralExport\CrossRefToXref to 0
- Set GeneralExport\DitaXrefElem to “” (nothing)
- Set GeneralExport\DitaLinkElem to “” (nothing)

RELATED INFORMATION:

"Using the Reference Manager" on page 18
"Xref to Hyperlink" on page 14

Creating Element Templates

Provides a method to add predefined structure and content to new files.

An element template is an FM binary file that contains structural and textual content that is inserted into a new topic when it is created. You can have many different element templates for each topic type. Available element templates may be selected in the New DITA File dialog.

By default, element templates are stored in the folder that contains the structure application template file. You can specify another folder (local or network location) to store these files in the New File Options dialog which is available from the New DITA File dialog or the Options dialog.

Element template files must follow a strict naming convention in order to appear in the list in the New DITA File dialog. The file name pattern is *new~<topictype>~<template name>.fm*. That is “new” followed by a tilde followed by the topic or map element type (such as “topic”, “concept”, “task” etc.), followed by a tilde, followed by the name that you want to appear in the list, with an “.fm” file extension.

To create an element template just create a new DITA file and insert the elements and content that you want in the template, then save this file to a binary FM file into the specified element template folder. Note that when the new file is created using your template, the text of the title element will be replaced with the text entered in the New DITA File dialog, and the ID attribute of the root topic will be replaced with a new unique ID (assuming that the “auto-add IDs” option is enabled in Options).

Sample topic and task element templates are provided in the DITA-FMx Topic application folder.

RELATED INFORMATION:

"New DITA File" on page 1

Switching Between DITA-FMx and FM-DITA

If needed, you can easily switch between DITA-FMx and the built-in FM-DITA support.

For testing purposes, you may want to alternate between DITA-FMx and the built-in FrameMaker DITA support. Fundamentally, this is very simple to do, and if you do this frequently, there are things you can do to make it even easier.

All that is required to switch between these two DITA plugins is to update the *maker.ini* file. When you installed DITA-FMx, you should have commented out the lines that initialize the default DITA support. To enable the default support and disable DITA-FMx, just uncomment the “ditafm” lines and comment out the “ditafmx” lines. Then to switch back to DITA-FMx just do the reverse. Of course, each time you modify the *maker.ini* file you’ll ned to restart FrameMaker to initialize the alternate plugin.

IMPORTANT: *Before modifying the maker.ini file you should make a backup and keep it in a safe place.*

You may want to set up the *maker.ini* file so it’s easy to make this change. Just move the lines to the end of the APIClients section and add some comments to look like this:

```
; Default FM2019-DITA
;ditafm=Standard, Translation client for DITA, fminit\ditafm.dll, structured
;ditafm_app=Standard, Translation client for DITA, fminit\ditafm_app.dll, structured
;Dita OT=Standard, DITA OT client for DITA, fminit\openToolkit.dll, structured

; DITA-FMx
ditafmx=Standard, ditafmx, DITA-FMx-2\ditafmx_150.dll, structured
ditafmx_app=Standard, ditafmx_app, DITA-FMx-2\ditafmx_150_app.dll, structured
```

This way you can easily add or remove the semicolon to comment or uncomment the appropriate lines.

One additional step that will streamline this process is to change the FM-DITA entry for “ditafm_app” to “ditafmx_app,” then make the same change in the structure application definitions file (change “ditafm_app” to “ditafmx_app” in the UseApiClient entries). This won’t change the functionality of the FM-DITA import/export client, it just means that you can use the same structure application for both plugins and you won’t have to edit the structapps file each time you switch. If you call them both “ditafmx_app,” then when you install an update of DITA-FMx, it will be an easier installation.

Leximation provides a plugin called IniSwitcher that modifies specified lines in an INI file from a command within FrameMaker. This plugin was specifically developed to make it easy to switch between the DITA authoring plugins. If you are interested in using this plugin, check www.leximation.com/tools or contact Leximation for more information.

RELATED INFORMATION:

"Installing DITA-FMx for the First Time" on page 2

Switching Between DITA-FMx 1.1 and 2.0

Don't want to completely switch to DITA-FMx 2.0 yet? You can safely install FMx 2.0 along side of FMx 1.1 and switch back and forth.

DITA-FMx 2.0 installs into a *DITA-FMx-2* folder in the main FrameMaker program folder. Because of this, your DITA-FMx 1.1 installation will be left untouched after installing DITA-FMx 2.0.

After running the installer, your *maker.ini* file will have been updated to point to the new DITA-FMx 2.0 DLLs. If you'd like to be able to switch between both DITA-FMx versions, just edit the *maker.ini* file as shown below.

```
;==============================
; FMx 1.1
;ditafmx=Standard, ditafmx, DITA-FMx\ditafmx_110.dll, structured
;ditafmx_app=Standard, ditafmx_app, DITA-FMx\ditafmx_110_app.dll, structured

; FMx 2.0
ditafmx=Standard, ditafmx, DITA-FMx-2\ditafmx_110.dll, structured
ditafmx_app=Standard, ditafmx_app, DITA-FMx-2\ditafmx_110_app.dll, structured
;==============================
```

The example shown will enable DITA-FMx 2.0. To switch to DITA-FMx 1.1, just add semicolons in from of the two "2.0" entries and remove the semicolons in from of the "1.1" entries, then restart FrameMaker.

INI-Only Settings

Describes the settings that must be applied manually in the ditafmx.ini file.

The default values for these parameters should be sufficient for most people, but there are reasons that you may want to change the default behavior. The following settings must be changed manually in the *ditafmx.ini* file.

INIOnly section

FlattenedConrefAttr - Specifies an attribute name to set on conref elements that have been flattened during the book-build process. By default this is disabled (set to an empty string), but if you set this to an attribute name such as “outputclass”, that attribute will be set with the value of “flattened-conref”. This can be useful if you want to apply formatting to content that was previously a conref (but has been flattened).

MaxRefLevels - Specifies the number of nested files that are opened due to the auto-loading of references (xref or conref). If your files never have references within references (such as an xref within a conref), then you should set this to a value of “1”. If your files do make use of nested references, set this value equal to the maximum number of reference levels that exist. The greater the number the longer it may take to open files since all references in all opened files will resolve (unless limited by this option). Valid values are 0 through 9 or “*” (asterisk means unlimited levels). Setting this value to 0 disables the auto-updating of xrefs and conrefs. (Defaults to “2” if this parameter is missing or set to a null string.)

This reference resolving process is used when generating a book from a map, but is not used while authoring unless the INIOnly/UseRefList parameter is set to “0”.

UseRefList - Controls the way references are resolved while authoring. If set to “1” (the default), references (xrefs and conrefs) are resolved “on disk,” allowing for much faster processing and less time required to open files. If set to “0” all referenced files are opened in FrameMaker in order to resolve the references. The number of files opened is determined by the reference nesting level, which is controlled by the General/MaxRefLevels parameter. (Defaults to “1” if this parameter is missing or set to a null string.)

StructappsFile - Specifies an alternate file to use as the structure application definitions file. In order for this to function properly you must also have set the Directories/StructureDir parameter in the *maker.ini* file. (No default value, may be null.)

XrefElements - A vertical-bar delimited list of element names that defines the elements that are valid as xref or link targets. For example, setting XrefElements to the following string limits the xref targets to the elements specified:

- XrefElements=topic|concept|task|reference|section|dt

If this parameter is not set, all elements are valid as xref or link targets. (No default value, may be null.)

DitaFMxGuide - Specifies the name of the DITA-FMx Help file (relative to the *FrameMaker\DITA-FMx* folder, unless an absolute path is specified). This Help file may be a CHM file or an EXE, or it may specify a URL (via the “http:” or “file:” protocols). If an EXE is specified, this is likely an AIR Help file, but may be any executable Help application set up to accept a command line parameter that indicates the target HTML filename to display. (Defaults to “ditafmx.chm” if this parameter is missing or set to a null string.)

DitaHelpKeys - Specifies the shortcut keystrokes to launch context-help for DITA authoring (this runs the “DitaReference” Help file). The keys defined here use the syntax for shortcuts as specified in the FDK Reference. (Defaults to “~/F1”, Alt+F1, if this parameter is missing or set to a null string.)

DitaReference - Specifies the name of the DITA Reference Help file (relative to the *FrameMaker\DITA-FMx* folder), which is used for element-based context sensitive Help. This Help file may be a CHM file or an EXE, or it may specify a URL (via the “http:” or “file:” protocols). If an EXE is specified, this is likely an AIR Help file, but may be any executable Help application set up to accept a command line parameter that indicates the target HTML filename to display. (Defaults to “ditafmx-ref.chm” if this parameter is missing or set to a null string.)

Displaying the DITA Reference Help uses a combination of this and two other parameters, DitaRefTargetType and DitaRefTargetPath. If a DITA document is open and the insertion point is within a DITA element, when the user presses Alt+F1 (as defined by the DitaHelpKeys parameter), the following action is issued:

<DitaReference> [*<DitaRefTargetPath>*]*<element-tag><DitaRefTargetType>*

If the DitaReference parameter specifies a CHM or EXE file, the DitaRefTargetPath, selected element tag name, and the DitaRefTargetType (file extension) values are concatenated and passed to the Help system.

If the DitaReference parameter specifies a URL, the DitaReference, DitaRefTargetPath, selected element tag name, and the DitaRefTargetType (file extension) values are concatenated and the resulting URL is passed to the default web browser application.

DitaRefTargetType - Specifies the file extension of target topics in the “DitaReference” Help file for context sensitive Help. For additional information, refer to the DitaReference parameter above. (Defaults to “.html” if this parameter is missing or set to a null string.)

DitaRefTargetPath - Specifies an optional “path” to target topics in the “DitaReference” Help file for context sensitive Help. For additional information, refer to the DitaReference parameter above. (No default value, may be null.)

MapFromOutlineTemplate - Specifies the “Map from Outline” template file used as the template for the **New ‘Map from Outline’ Template** command. (Defaults to “$STRUCTDIR\xml\DITA-FMx_1.2\map-from-outline_template.fm” if this parameter is missing or set to a null string.)

BookTitleVariableName - Specifies the name of the variable that is updated with the book title (from map/title, map/@title, bookmap/title, or bookmap/booktitle/mainbooktitle). This variable is only updated in generated list files if it is included in the generated list template (structured files can access the book title via the attribute on the fm-ditabook element). (Defaults to "FMxBookTitle" if this parameter is missing or set to a null string.)

StripClassAttribute - Set this to "0" to prevent the @class attribute value from being deleted when changing an element's type (using the "Change" button in the Element Catalog). (Defaults to "1", enabled, if this parameter is missing or set to a null string.)

TplDelimChar - Set this to specify the character to use as the file name delimiter for new file templates (*new~<topictype>~<template name>.fm*), book-build component templates (*tpl~<mapelemtype>.fm*), and generated list templates (*gentpl~<mapelemtype>.fm*). If necessary you can set this to a character, such as the hash symbol, #. (Defaults to "~" if this parameter is missing or set to a null string.)

MaxOpenFiles - Specifies the number of XML files opened before a warning is displayed. This is convenience when working with FM7.2 and FM8 to allow time to restart FrameMaker before it crashes. These versions of FrameMaker have a bug that results in a crash after opening 300+ XML files in the same session. (Defaults to "300" if this parameter is missing or set to a null string.)

UseBooklistPlaceholder - Set this to "1" to revert to the pre 1.1.11 DITA-FMx method of requiring a placeholder file to generate a list. The default value "0" indicates that DITA-FMx will honor the "proper" syntax for frontmatter and backmatter list elements which does not require a placeholder file.

NewMapShortcuts and **NewTopicShortcuts** - Allows customization of the shortcut keys assigned to the New File menu items. The value for both of these is a space-delimited string of doctype/key pairs. Use the following syntax:

<doctype>:<key-sequence>

Where:

<doctype> is the root topic or map element.

<key-sequence> is the key sequence for the shortcut. To specify an ESC sequence the string starts with "\!" (backslash, exclamation). This is followed by 2 or 3 characters or numbers (case sensitive). (In theory you can also use ALT "~" or CTRL "^" or SHIFT "+", but this may or may not work well. You can also "try" to use function keys, which are defined as slash followed by "F" and the number, as in "/F3".)

For example, to create shortcuts for map, bookmap, topic, and task, you could use the following:

```
NewMapShortcuts=map:\!NMM bookmap:\!NMB
NewTopicShortcuts=topic:\!NTO task:\!NTA
```

FilteringAttributes - Specifies the attributes that are considered when performing ditaval filtering. The default value is a space-delimited string of attribute names: `audience product platform otherprops rev props`.

UseDatatype - As of v.2.0.06, graphic overlay object type information is now stored in the <data> element's @name attribute. If you want to continue using the @datatype attribute, set this parameter to "1".

NavtitleLen - As of v.2.0.06, controls the length of the content in <navtitle> element when updated based on the referenced topic title text. This defaults to 250 characters (bytes), which should be more than sufficient, but for multibyte content, you may want to increase this value.

AutoXrefTextElems - As of v.2.0.07, specifies a space delimited list of elements to automatically add alt text to xrefs based on the target element type selected in the Reference Manager. If not set, no automatic alt text is added.

ReportRO - As of v.2.0.07, if set to "0", disables the notification when opening a read-only file. If not set, this notification is enabled.

INIOnly section (for use with the Set Attributes command)

SetAttrIgnore - Specifies the attributes to ignore (and not display) in the Set Attributes dialog. This is a vertical bar delimited string. By default this is set to the following values, you can change these values or set this to a null string as needed.

```
xtrc|xtrf|xmlns|class|xmlns:ditaarch|ditaarch:DITAArchVersion|domains
```

SetAttrStrings - Specifies the "filtering groups" INI file which defines values for the specified "Strings" attributes. These values are organized into named groups which can be associated with a specific file system path. All topic files that fall within the specified root path are mapped to that group. Each group lists one or more attributes and a vertical bar delimited list of default values for that attribute. This lets you provide alternate filtering options for different projects that use the same structure application without modifying the EDD. (Defaults to "FilterGroups.ini" if this parameter is missing or set to a null string.)

Unless the value of the SetAttrStrings parameter specifies an absolute path and file name, it is relative to the user's DITA-FMx folder. If you use a SetAttrStrings file, no default values need be set in the EDD at all. This adds flexibility to the EDD not otherwise obtainable. For more information and details, see Set Attributes.

SetAttrStringsDefault - Specifies the "default" value that is ignored when displaying the list of defaults.

NOTE: This INI parameter is for very specialized and atypical use. Unless you are doing special FDK processing using attributes of type "Strings" you can safely ignore the following information.

When you specify default values to a Strings attribute in the EDD, the first default may be used as the actual default for certain types of processing. If you want to specify a special value as the first default so that your processing can match on it and ignore the value, enter that string as the value for the SetAttrStringsDefault parameter. A likely value is the dot (".") character, but you can specify any value that makes sense for your processing. If you specify this value, it will be ignored and excluded from the options displayed in the Set Attribute dialog. (Defaults to "." if this parameter is missing, but this value may be set to a null string.)

If you do not have processing that cares about the initial default value of a Strings attribute, you can ignore this parameter.

INIOnly section (for use with content management systems)

CMSClientName - Specifies the FDK client name used to connect with the CMS bridge or connector. Typically set automatically by the CMS client. (No default value, may be null.)

CMSImageDefaultDpi - Specifies the default DPI for new images. Currently only used for XDocs CMS. (Defaults to "72" if this parameter is missing or set to a null string.)

GeneralImport section

IndextermProcessing - If set to 1, enables conversion of indexterm elements to proper format required by FrameMaker. This parameter is set through the user interface as the Indexterm to Fm-indexterm Conversion option in the DITA-FMx Options dialog and should not be set manually without care. (Defaults to "1" if this parameter is missing or set to a null string.)

StripPadding - If set to 1, strips leading space from the XML. This parameter is set through the Normalize Whitespace on Import option in the DITA-FMx Options dialog. (Defaults to "1" if this parameter is missing or set to a null string.)

StripTrailing - If set to 1, strips space characters from the XML that follow the opening or closing block-level element tags. This only strips the spaces when there are no other non-space characters in the node that follow the open or close element. (Defaults to "1" if this parameter is missing or set to a null string.)

TableProcessing - If set to 1, enables the counting of table columns for proper import of simpletable and other table types into FrameMaker. (Defaults to "1" if this parameter is missing or set to a null string.)

MissingImageDir - Specifies the folder that the "missing image" files are stored. This location is relative to the DITA-FMx folder in the Program

Files area, and defaults to “missing-images” if not specified. To specify an alternate location, specify an absolute path to the folder that contains these files. To disable this feature, set this parameter to “0” (zero) or “Off”.

This feature is intended to eliminate the XML parser log messages during a book-build that may incorrectly report images as missing. This would often happen when generating a book file into a folder structure that varies significantly from that of the source folder structure. On import, the images would in fact not be found, but the DITA-FMx “Reload References” book-build option would resolve these references.

If enabled, during the book-build process the images in the MissingImageDir folder will temporarily replace the actual images to prevent the errant XML parser log messages. This folder must contain a “break” and “inline” image for each file type in use in your documents. These files are named “missing-image-break.<EXT>” and “missing-image-inline.<EXT>”.

GeneralExport section

IndextermProcessing - If set to 1, enables conversion of indexterm elements from the format required by FrameMaker back to one required by DITA. This parameter is set through the user interface as the Indexterm to Fm-indexterm Conversion option in the DITA-FMx Options dialog and should not be set manually without care. (Defaults to “1” if this parameter is missing or set to a null string.)

XrefProcessing - If set to 1, enables processing and conversion of xrefs and links. (Defaults to “1” if this parameter is missing or set to a null string.)

CrossRefToXref - If set to 1, enables the conversion of FM-based cross-references to DITA-based xrefs and links. (Defaults to “1” if this parameter is missing or set to a null string.)

DitaXrefElem - Defines the element name to use when mapping FM-based cross-refs to DITA xrefs on file export (only used if GeneralExport/CrossRefToXref is enabled). (Defaults to “xref” if this parameter is missing or set to a null string.)

DitaLinkElem - Defines the element name to use when mapping FM-based cross-refs to DITA links on file export (only used if GeneralExport/CrossRefToXref is enabled). (Defaults to “link” if this parameter is missing or set to a null string.)

IgnoreElemPrefix - If you use elements in FrameMaker that do not exist in the DITA DTDs, you should make sure they all have the same prefix (“fm-” for example). This INI setting specifies that prefix. This is required for proper generation of conrefs on export, and should be used in conjunction with any read/write rules that may be needed. (Defaults to “fm-” if this parameter is missing or set to a null string.)

WriteLinkTextToFmXrefs - If set to 1, specifies that the link text of fm-xrefs is written to the xref element on file save. Normally, this link text is regenerated based on the target element text, but if you want the FM-generated text to be available for other processing engines, this should be enabled. (Defaults to “0” if this parameter is missing or set to a null string.)

WriteLineBreakPIs - If set to 1, specifies that line breaks (Shift+Enter) are written to the XML file as a processing instruction (PI). If DITA-FMx encounters a PI of `<?dtall break="line"?>`, it inserts a line break, thus allowing proper round-tripping of line breaks between authoring and publishing. Note that the line breaks will only be persevered when publishing through FrameMaker (and DITA-FMx), and other tools that honor this processing instruction. (Defaults to “1” if this parameter is missing or set to a null string.)

WriteDictionaryPIs - If set to 1, specifies that words flagged by the spelling checker as “Allow in Document” are written to the XML file as a processing instruction (PI) before the root topic node (`<?dtall dict="allowed-word"?>`). On file open these words are loaded in to the document dictionary for the current file. Note that these processing instructions are not likely to be honored by other tools. (Defaults to “1” if this parameter is missing or set to a null string.)

NOTE: *Over time if you clone a topic file for other uses, these processing instructions may build up in the file. You can delete the items in the document dictionary for a given file by opening that file and choosing the Dictionaries button in the Spelling Checker dialog. In that dialog select the Document Dictionary option, choose Clear from the list, then choose OK.*

FmXrefUseOutputclass - If set to 1, specifies that <fm-xref> elements use the @outputclass attribute rather than the @type attribute to store the cross-ref format name. (In DITA-FMx 1.1 the default was “0”, but now it is “1”.)

FmDpiUseOutputclass - If set to 1, specifies that the “fmdpi” value is assigned to the @outputclass attribute rather than the @otherprops attribute.

DITA-FMx 1.1 used the @otherprops attribute, which was not the intended use for that attribute. Using the @outputclass attribute is more in line with the DITA specification. This setting controls how this attribute is added to new elements, but does not affect existing elements. The sizing of images will work properly regardless of the attribute the “fmdpi:*NN*” value is assigned.

TableImport section

SetColumnsProp - If set to 1, enables the automatic assignment of the “columns” property. (Defaults to “1” if this parameter is missing or set to a null string.)

SetColumnWidthsProp - If set to 1, enables the automatic assignment of the “column width” property. (Defaults to “1” if this parameter is missing or set to a null string.)

CustomTableCount and ***N*** - Specifies the number of instances of simpletable elements that have been specialized that need to be automatically analyzed for their column number. The *Count* value should match the number of *N* values. Each *N* value should have the following format: “<table-element>|<row-element>|<cell-element>”.

IndexOptions section

IndexMarkerType - Specifies the name of the index marker type. Useful for Japanese systems, which use a localized name for the index marker. (Defaults to “Index” if this parameter is missing or set to a null string.)

ExternalApplications section

EPUBReaderAppExe - Registers the path and filename of the AZARDI EPUB reader application to use if an EPUB file is specified in the DitaFMxGuide or DitaReference parameters. (This feature is considered experimental.)

BuildFile section

AntCommand - Specifies the command or path/command used to run Ant. The default is “ant”, but if you need to specify another executable or if you need to include the path, change this value. This value is used for both Generate Output options, Current File and Selected Target. (Defaults to “ant” if this parameter is missing or set to a null string.) This parameter is ignored for OT 2.x or later.

EnvironmentSetup - This parameter can be set through the DITA-FMx Options dialog in the External Application Settings dialog. (No default value, must be set to use the Generate Output command.)

For OT 2.x or later, specifies the *bin\dita.bat* file in the selected DITA-OT installation.

For OT 1.x, specifies the batch file to run before the Ant script in order to set up any needed environment variables.

DitaDir - Specifies the folder that contains the DITA-OT installation. This parameter can be set through the DITA-FMx Options dialog in the External Application Settings dialog. (No default value, must be set to use the Generate Output command.)

EnvironmentSetup-<buildname> - Specifies an alternate batch file to override the default specified by EnvironmentSetup. Allows you to define builds that rely on different DITA environments.

DitaDir-<buildname> - Specifies an alternate location for the DITA-OT folder to override the default specified by DitaDir. Allows you to define builds that rely on different DITA environments.

AntScript - Specifies the filename of the Ant script that is run for the Generate Output: Current File option. The default is “ditafmx-ant.xml”, but if you need to specify another filename, change this value. This file-name is assumed to be relative to the path specified by the *DitaDir* parameter. (Defaults to “ditafmx-ant.xml” if this parameter is missing or set to a null string.) This parameter is ignored for OT 2.x or later.

Count and ***N*** - Specifies the number of build options available in the *AntScript* file (and thus displayed in the Generate Output: Current File output option). If you modify the targets in that script, be sure to update the *Count* value and add/remove the related *N* value.

AntBuild section

Count and ***N*** - Specifies the number of “ANT:” sections which define the available build options displayed in the Generate Output: Selected Target output option. For each “ANT:” section added, you must update the *Count* value and add/remove the related *N* value. The value of each *N* parameter must exactly match the text following the “ANT:” section name.

ANT:<buildname> section

BuildFile - Specifies path and filename of the associated Ant script (use double backslashes as the directory delimiter). (No default value, must be set to use the Selected Target option of the Generate Output command.)

EnvironmentSetup - Specifies a batch file to run before the Ant script in order to set up any needed environment variables.

Target - Specifies target within the *BuildFile*. (No default value, must be set to use the Selected Target option of the Generate Output command.)

OutputDir - Specifies the path to the output directory (use double backslashes as the directory delimiter). (No default value, must be set to use the Selected Target option of the Generate Output command.)

Logfile - Specifies the path and filename to the log file (use double backslashes as the directory delimiter). (No default value, must be set to use the Selected Target option of the Generate Output command.)

RELATED INFORMATION:

"DITA Options" on page 33
"Tips and Troubleshooting" on page 14
"Using the Reference Manager" on page 18
"Working with Images" on page 27
"Working with Tables" on page 35

Uninstalling DITA-FMx

To remove DITA-FMx and put things back to the original form.

There is no Uninstall application provided with DITA-FMx. Because most of the installation is done manually, it would be misleading and possibly harmful to provide an uninstaller.

To remove DITA-FMx, just delete the *DITA-FMx-2* folder and remove the “ditafmx” lines from the APIClients section of the *maker.ini* file. If you just want to disable DITA-FMx, you can comment out the “ditafmx” lines by adding a semicolon at the beginning of each line.

In general, to reinstall the default DITA support, just uncomment the lines in the *maker.ini* file that you commented out when installing DITA-FMx. If you deleted those lines, just add the following lines to the end of the APIClients section in the *maker.ini* file.

FrameMaker 8:

```
ditafm=Standard, Translation client for DITA, fminit\ditafm.dll, structured
ditafm_app=Standard, Translation client for DITA, fminit\ditafm_app.dll, structured
ditabook=Standard, Translation client for DITA, fminit\ditabook.dll, structured
```

FrameMaker 9/10/11:

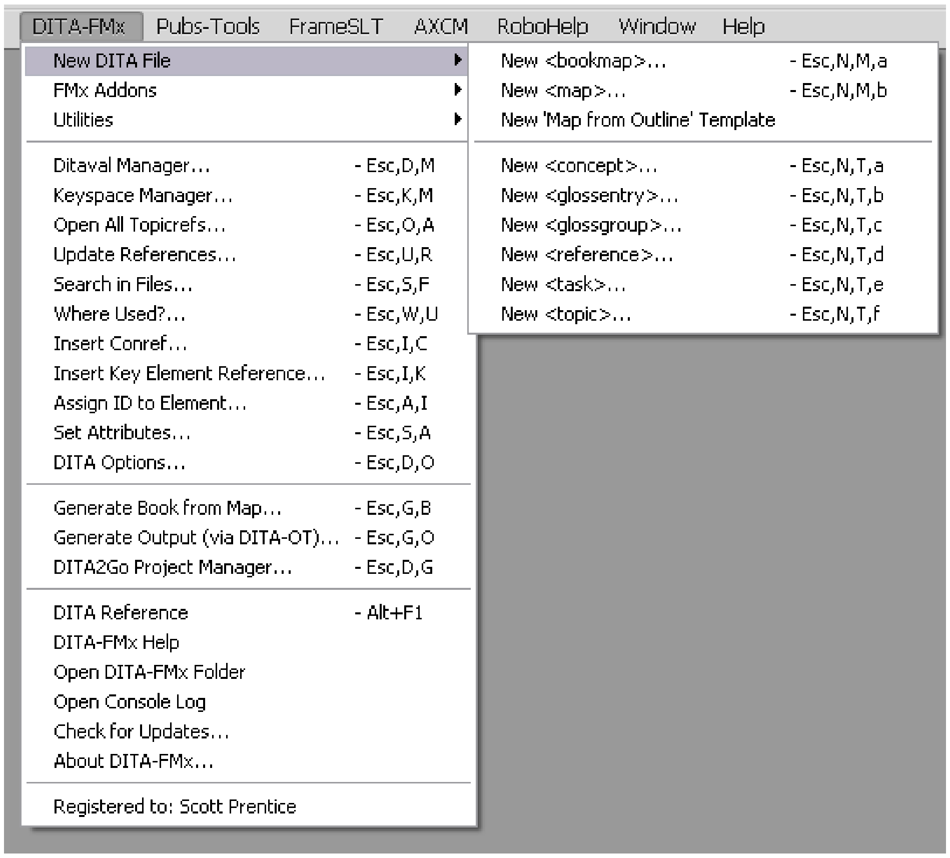
```
ditafm=Standard, Translation client for DITA, fminit\ditafm.dll, structured
ditafm_app=Standard, Translation client for DITA, fminit\ditafm_app.dll, structured
```

FrameMaker 12/2015/2017/2019/2020:

```
ditafm=Standard, Translation client for DITA, fminit\ditafm.dll, structured
ditafm_app=Standard, Translation client for DITA, fminit\ditafm_app.dll, structured
Dita OT=Standard, DITA OT client for DITA, fminit\openToolkit.dll, structured
```

3 DITA-FMx Commands

Describes the commands and functionality available in DITA-FMx.


DITA-FMx
Pubs-Tools
FrameSLT
AXCM
RoboHelp
Window
Help
New DITA File
FMx Addons
Utilities
Ditaval Manager... - Esc,D,M
Keyspace Manager... - Esc,K,M
Open All Topicrefs... - Esc,O,A
Update References... - Esc,U,R
Search in Files... - Esc,S,F
Where Used?... - Esc,W,U
Insert Conref... - Esc,I,C
Insert Key Element Reference... - Esc,I,K
Assign ID to Element... - Esc,A,I
Set Attributes... - Esc,S,A
DITA Options... - Esc,D,O
Generate Book from Map... - Esc,G,B
Generate Output (via DITA-OT)... - Esc,G,O
DITA2Go Project Manager... - Esc,D,G
DITA Reference - Alt+F1
DITA-FMx Help
Open DITA-FMx Folder
Open Console Log
Check for Updates...
About DITA-FMx...
Registered to: Scott Prentice
New <bookmap>... - Esc,N,M,a
New <map>... - Esc,N,M,b
New 'Map from Outline' Template
New <concept>... - Esc,N,T,a
New <glossentry>... - Esc,N,T,b
New <glossgroup>... - Esc,N,T,c
New <reference>... - Esc,N,T,d
New <task>... - Esc,N,T,e
New <topic>... - Esc,N,T,f

RELATED INFORMATION:

"Using DITA-FMx" on page 1
"Installation and Setup" on page 1
"Extending DITA-FMx" on page 1

New DITA File

Creates a new DITA map or topic file.

There are two groups of items on the **DITA-FMx > New DITA File** menu. Items above the divider create a new map file and items below the divider create a new topic file. The number and label text of these items will vary depending on the map and topic structure applications you have selected in the Options dialog. Any element definition that defines a class attribute with a default value that contains “map/map” will be considered a “map” and any with a default value that contains “topic/topic” are considered a “topic.”

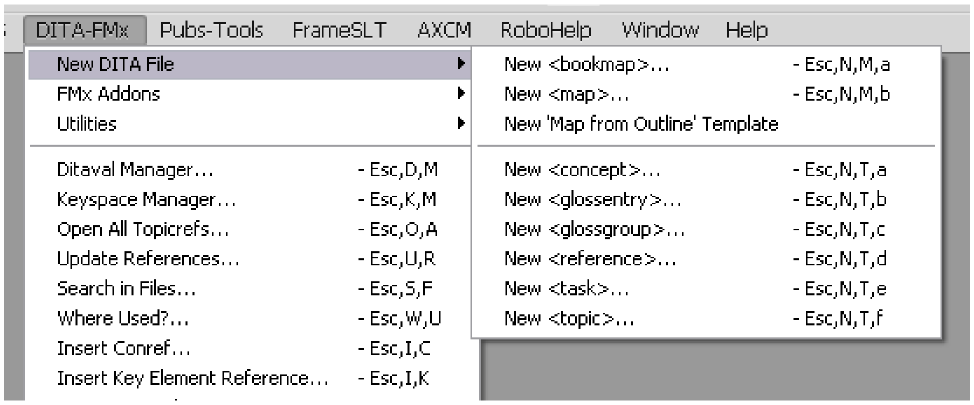
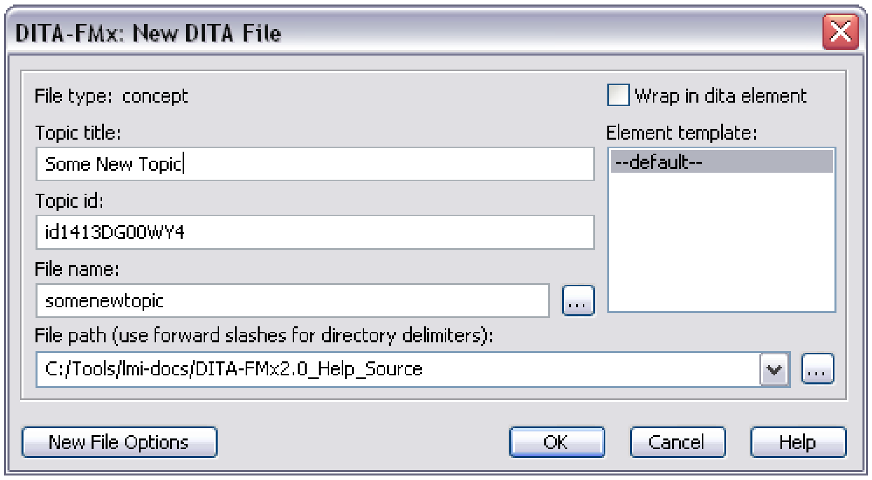

DITA-FMx
Pubs-Tools
FrameSLT
AXCM
RoboHelp
Window
Help
New DITA File
FMx Addons
Utilities
Ditaval Manager... - Esc,D,M
Keyspace Manager... - Esc,K,M
Open All Topicrefs... - Esc,O,A
Update References... - Esc,U,R
Search in Files... - Esc,S,F
Where Used?... - Esc,W,U
Insert Conref... - Esc,I,C
Insert Key Element Reference... - Esc,I,K
New <bookmap>... - Esc,N,M,a
New <map>... - Esc,N,M,b
New 'Map from Outline' Template
New <concept>... - Esc,N,T,a
New <glossentry>... - Esc,N,T,b
New <glossgroup>... - Esc,N,T,c
New <reference>... - Esc,N,T,d
New <task>... - Esc,N,T,e
New <topic>... - Esc,N,T,f

Figure 3-1: New DITA File menu

Each topic or map type is assigned a keyboard shortcut, if you want to define your own shortcut for specific topic types, see the information on the NewMap-Shortcuts and NewTopicShortcuts parameters in INI-Only Settings.

At the bottom of the “map” area is the command **New ‘Map from Outline’ Template**. This command creates a new FM file using the “Map from Outline” file as the template. By default, this template is named *map-from-outline_template.fm* and is in the root of the DITA-FMx structure application folder ($STRUCTDIR\xml\DITA-FMx_1.2\). You can specify an alternate template file by changing the MapFromOutlineTemplate parameter in the INIOnly section of the *ditafmx.ini* file. For more information, see Build Map from Outline and INI-Only Settings.

Choosing one of the **New <ELEMNAME>** menu items displays the New DITA File dialog. This dialog provides fields for entering the topic or map title, the topic ID, the file name, file path, and an optional element template to insert predefined structure and content into the new file.


DITA-FMx: New DITA File

File type: concept

Wrap in dita element: ☐

Topic title: Some New Topic

Topic id: id1413DG00WY4

File name: somenewtopic

File path (use forward slashes for directory delimiters): C:/Tools/lmi-docs/DITA-FMx2.0_Help_Source

Element template: --default--

Figure 3-2: DITA-FMx New DITA File dialog box

For new topics, the title is inserted into the first title element, for maps, the title text is entered as the map/@title attribute or in the map/title element (depending on the DITA version). The file name field is populated based on the value of the New File Name Format setting (in Options: New File Options). If the new file name format includes a building block that includes the title, the file name field will update as you enter text into the title field.

Entering a folder name into the File Name field will create the new file in the specified folder. If that folder, or folders, do not exist, you will be prompted to allow them to be created.

TIP: *You can use building blocks to automatically create new files in folders based on the topic type. For an example, see the New File Options topic.*

You can also select the **Wrap in Dita Element** option to create a topic file that includes the dita element as the root element. This is useful if you are creating files with nested topics, and required if you want to have multiple top-level sibling topics. The default value for this option is defined in the **New File Options** dialog.

When you choose OK, the file is immediately saved as XML using the name specified. If you don't provide a file name extension for the new topic file, one is added based on the type specified in the Default File Type option in the Options dialog. DITA map files are always given the extension of "ditamap."

When creating a new topic or map file, if your insertion point is in a DITA map at a location that is valid for a topicref-based element, a dialog displays asking for the type of element to insert, or none (to not insert as an element).

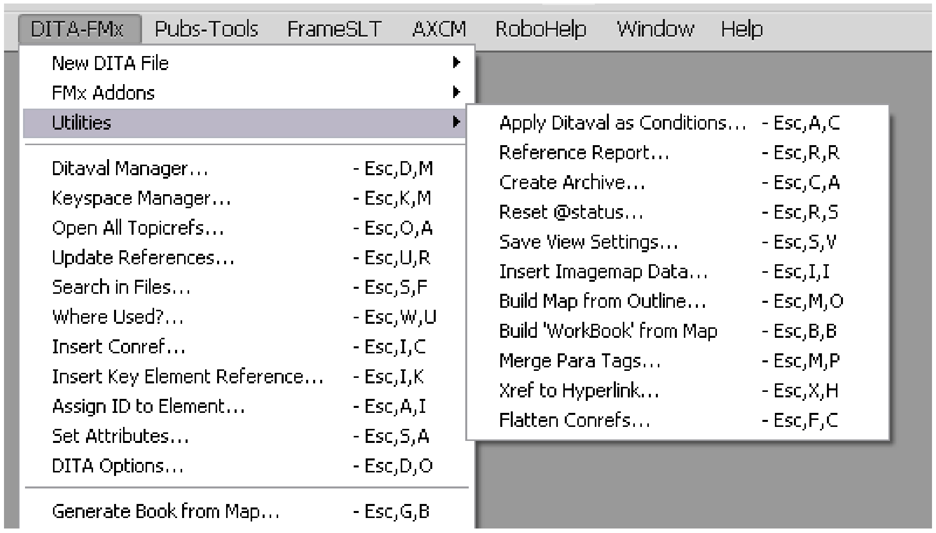
Creation of new DITA files differs from the standard FrameMaker "New" command because FrameMaker does not provide a method for creating an "Untitled" XML file (one that is not saved to the file system). If you want to start with an "Untitled" file, select the standard New command (**File** > **New**) and use the DITA template as the template file.

Related information:

"Creating Element Templates" on page 49
"New File Options" on page 41
"Auto-Prolog Options" on page 44
"Build Map from Outline" on page 11
"INI-Only Settings" on page 51

Utilities

Less frequently used commands.


DITA-FMx
Pubs-Tools
FrameSLT
AXCM
RoboHelp
Window
Help
New DITA File
FMx Addons
Utilities
Ditaval Manager... - Esc,D,M
Keyspace Manager... - Esc,K,M
Open All Topicrefs... - Esc,O,A
Update References... - Esc,U,R
Search in Files... - Esc,S,F
Where Used?... - Esc,W,U
Insert Conref... - Esc,I,C
Insert Key Element Reference... - Esc,I,K
Assign ID to Element... - Esc,A,I
Set Attributes... - Esc,S,A
DITA Options... - Esc,D,O
Generate Book from Map... - Esc,G,B
Apply Ditaval as Conditions... - Esc,A,C
Reference Report... - Esc,R,R
Create Archive... - Esc,C,A
Reset @status... - Esc,R,S
Save View Settings... - Esc,S,V
Insert Imagemap Data... - Esc,I,I
Build Map from Outline... - Esc,M,O
Build 'WorkBook' from Map - Esc,B,B
Merge Para Tags... - Esc,M,P
Xref to Hyperlink... - Esc,X,H
Flatten Conrefs... - Esc,F,C
Figure 3-3: Utilities menu

Related information:

"Merge Para Tags" on page 13
"Xref to Hyperlink" on page 14
"Flatten Conrefs" on page 15
"Apply Ditaval" on page 4

Apply Ditaval

Applies filtering to files based on the properties defined in a ditaval file.

The **Apply Ditaval** command offers two modes for ditaval filtering:

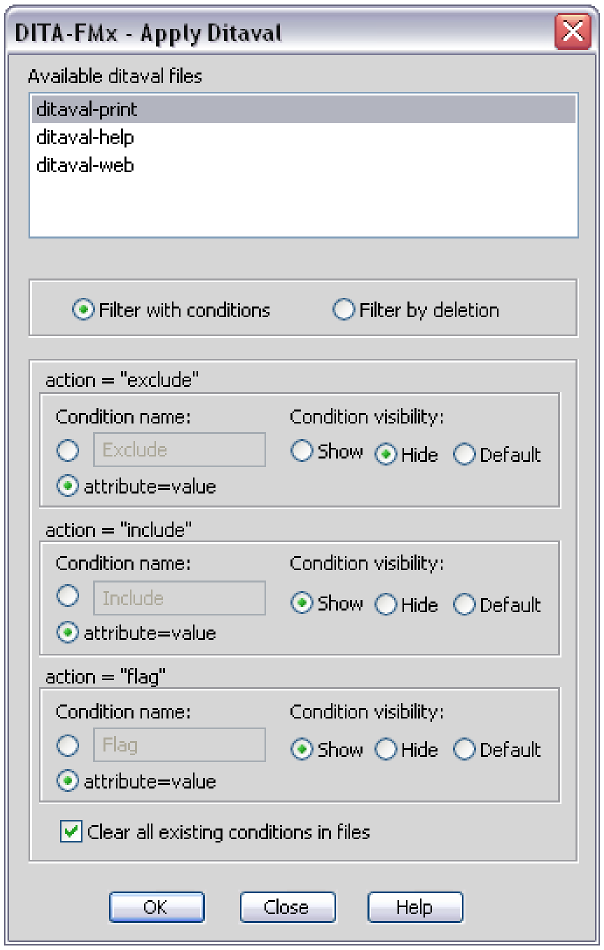
- **Filter with Conditions** - Applies filtering by mapping the ditaval properties to FrameMaker conditional tagging based on the options selected in the **Apply Ditaval** dialog. Due to the nature of FrameMaker conditions, for complex filtering, this option may not always result in the expected outcome.

- **Filter by Deletion** - Applies filtering by deleting “excluded” content as defined in the selected ditaval file. The result of using this option will often be closer to that of the filtering applied by the DITA-OT.

This command can be used to filter a topic while authoring or for filtering content that has been generated through the **Generate Book from Map** command. If the command is run on a topic file, the filtering is applied to that file. If a book is active, the filtering is applied to all files in that book.

In order to use this command you must have at least one ditaval file registered with DITA-FMx. A ditaval file is “registered” with DITA-FMx when you use the Ditaval Manager to create new ditaval files or add existing ditaval files. When you run the **Apply Ditaval** command, the dialog box lists the available ditaval files (those that have been registered).

IMPORTANT: *When applying filtering to the active topic file, keep in mind that the Filter by Deletion option actually deletes content from the file. Because of this it’s recommended that you only run this command on a duplicate file. When using the Filter with Conditions option on the active topic file, the conditional tagging is typically saved to the XML file as processing instructions (PIs). This may also not be a desirable situation. Use this command with caution on active topic files.*


DITA-FMx - Apply Ditaval

Available ditaval files
- ditaval-print
- ditaval-help
- ditaval-web

[✓ Filter with conditions] Filter by deletion

action = "exclude"
Condition name: Exclude [✓ attribute=value]
Condition visibility: Show [✓ Hide] Default

action = "include"
Condition name: Include [✓ attribute=value]
Condition visibility: [✓ Show] Hide Default

action = "flag"
Condition name: Flag [✓ attribute=value]
Condition visibility: [✓ Show] Hide Default

[✓ Clear all existing conditions in files]

Figure 3-4: DITA-FMx Apply Conditions dialog box

Filtering by deletion

When using the Filter by Deletion option, the rest of the options in the dialog are disabled; they only apply to the filter with condition option. The Filter by Deletion option deletes all elements with filtering attributes that match those specified with “exclude” <prop> values in the ditaval file.

Filtering with conditions

When using the Filter with Conditions option, the properties assigned to the ditaval @action values (“exclude”, “include” and “flag”). The settings in the three “action=” areas define the condition name and the visibility of the conditions that are applied when a <prop> @action attribute matches the specified type.

For each <prop> element in a ditaval file the @action attribute specifies either to exclude, include, or flag elements with matching attributes and values. When a ditaval file is applied as conditions in a FrameMaker file, elements are matched based on the @att and @val attributes, and a named condition is applied to each element. In the **Apply Conditions** dialog you specify that the condition name is defined as a fixed string (such as “Exclude”, “Include” or “Flag”) or that it is defined based on a combination of the @att and @val attribute values. For example, given the following line from a ditaval file, if the condition name is defined by the @att and @val attributes, it would be “audience=admin.”

```
<prop att="audience" val="admin" action="exclude" />
```

The condition visibility options are Show, Hide, and Default. If set to Show or Hide, the conditions will be shown or hidden accordingly. However, if Default is selected, the condition visibility will be defined by the settings in the template assuming that the condition is already defined.

This can also be done automatically during the Map to Book conversion process by selecting the **Apply Ditaval** option in the **Book Build Settings** dialog found in the **DITA Options** dialog.

NOTE: *The* ***Apply Ditaval as Conditions*** *command does not necessarily result in conditional filtering that matches that of the DITA Open Toolkit. The conditions are applied properly based on the filtering attribute values, but the default hide/show logic of FrameMaker conditions is not the same as the used by the OT. It may be possible to achieve this filtering through the use of Boolean conditional expressions.*

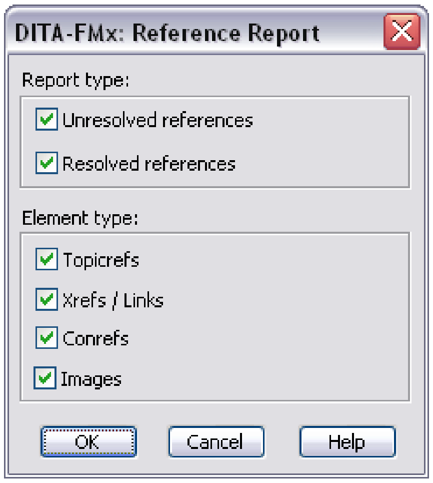
RELATED INFORMATION:

"Ditaval Manager" on page 18
"Generate Output" on page 78
"Setting Up Filtering Groups" on page 32
"Using the Apply Ditaval Command to Filter with Conditions" on page 26

Reference Report

Generates a report listing all resolved or unresolved references in the current file or map.

The report is generated as an FM file with two tables (one for resolved and the other for unresolved references). These tables can be sorted as needed using the standard table sorting fetures in FrameMaker. The document is locked (view only) and file references are hyperlinked.



- [✓] Unresolved references
- [✓] Resolved references
- [✓] Topicrefs
- [✓] Xrefs / Links
- [✓] Conrefs
- [✓] Images

Figure 3-5: DITA-FMx Reference Report dialog box

Related information:

"Update References" on page 21

Create Archive

Creates a ZIP archive of the current file and all referenced files.

The Create Archive command uses the command line syntax specified in the Archive field of the DITA Options: External Applications dialog to create a ZIP archive of all files referenced by the current file or map. This is an ideal way to create an archive of a project at a given point in time or makes it easy to package up a project to hand off to others.

When the command completes, a dialog displays the name of the archive file created. The archive generated is named *<filename>_zip.zip*, where <filename> is the root file name of the current file when the Create Archive command is used. For example, if the archive is created from the *project_a.ditamap* file, the archive created will be named *project_a_zip.zip*.

If you'd like to include additional files in the archive that are not specifically referenced (such as ditaval files), you can create an "archive baggage" file. When the Create Archive command runs, it checks in the folder of the current file for a file named *<filename>-archive.txt* and if this file is found, it includes the files listed in the new archive. For example, if the archive is created from the *project_a.ditamap* file, the command checks for a file named *project_a-archive.txt*. This TXT file should list each baggage file on a separate line. The file names are assumed to be relative to the current folder.

By default DITA-FMx uses an open source archive utility called "Info-ZIP." The default file name of this utility has been renamed to *ditafmx-zip.exe* and is installed to the DITA-FMx installation folder (in the Program Files area). You can specify an alternate archive utility or use different command line options, by changing the command line syntax in the DITA Options: External Applications dialog.

NOTE: *If the archive is not created, verify that the Command Line Syntax value in the DITA Options: External Applications dialog is valid. Deleting this value will reset it to the default value. You can test by running the ~tmpzip.bat batch file in the user's DITA-FMx folder (**DITA-FMx > Open DITA-FMx Folder**); run this from a command shell to see any errors that may display.*

Related information:

"External Application Settings" on page 56

Show Status

Applies conditional tagging to content based on the value of the status attribute. The conditional tagging in turn can apply special styling like change bars and strikethrough.

When run on a file or book, applies special status-oriented conditions to elements based on the value of the status attribute. The following list shows the condition names applied for each status value and the corresponding default style:

- "new" -> DITA-StatusNew (style: change bar)
- "changed" -> DITA-StatusChanged (style: change bar)
- "deleted" -> DITA-StatusDeleted (style: strikethrough)
- "unchanged" -> DITA-StatusUnchanged (style: overline)

If these conditions exist in the current file (as defined by the corresponding template), the styling applied is based on those condition definitions. If the conditions don't exist, they are created and asigned the styles listed above.

Related information:

"Reset Status" on page 9
"Authoring Options" on page 47

Reset Status

Deletes the value of the status attribute on all elements in the current book or file.

The "Set @status for new/changed elements" option (in the Options dialog) automatically sets the value of the status attribute on new and changed elements. Use this command when you need to reset this value (to nothing).

Related information:

"Show Status" on page 8
"Authoring Options" on page 47

Save View Settings

Preserves the user's view settings so they can restored without editing the default template file.

This command saves the values of the following view settings:

- Document zoom
- Attribute display options
- Element boundary type and view settings
- Visibility of borders, text symbols, rulers, and grid lines

To automatically restore these view settings to other documents as they are opened, enable the Restore Saved View Settings option.

The view settings are saved in the FM document, not in XML files, so if you change a view setting, then close and reopen a file, those settings are not reapplied unless this option is enabled. Without this feature, you would need to edit the structure application template file by setting the desired options and saving that template. This feature allows multiple people to use the same structure application without needing to make modifications for personal use.

NOTE: *This command does not save all of the possible "view" settings, just those listed above. To change other view options such as display or font units or grid spacing, edit those settings in the appropriate structure application's template file.*

Related information:

"DITA Options" on page 33

Insert Imagemap Data

Imports map data created for HTML imagemaps into your DITA topics.

If your documents make use of imagemaps, you can use this command to import the map data created by third party imagemap editors. You are prompted to select an XHTML imagemap data file (this file must be valid XHTML). The <area> elements from the first <map> element in the file is read and converted into the proper DITA structure, then inserted into the selected <imagemap> element in your document.

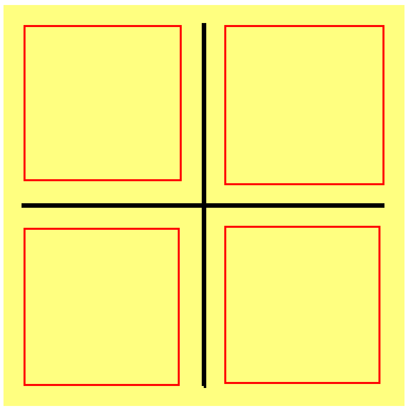
The imported data is not validated, and is inserted as defined in the selected file. At this time only the “rect” shape is support by DITA-FMx.

After importing the data, you can visualize the regions with the **Test Hotspots** command. This command is keyboard-only and is run by selecting the <imagemap> and pressing the shortcut keys **Esc,T,H**. This creates red text frames over the image based on the specified coordinates. For the “rect” shape, the coords value assumes the syntax of “*x1,y1,x2,y2*” (upper left and lower right corners, measured from the upper left corner of the anchored frame).

The @href values are assumed to be external references (URLs) and are converted into <xref> elements with @scope set to “external”. The following code sample shows the expected XHTML format:

```
...
<map name="mymap">
  <area shape="rect" coords="10,10,90,90" href="http://www.adobe.com"/>
  <area shape="rect" coords="110,10,190,90" href="http://www.leximation.com"/>
  <area shape="rect" coords="10,110,90,190" href="http://www.google.com"/>
  <area shape="rect" coords="110,110,190,190" href="http://www.yahoo.com"/>
</map>
...
```

The following image shows the result of applying this map to a simple image. The red boxes indicate the “hot” regions:



The coordinates of the hot spots (text frames) are aligned using a DPI of 72. If your image requires a different DPI, set the @outputclass attribute of the <image> element to “fmdpi:*NNN*” (where *NNN* is the DPI value). If you’ve enabled the “fmdpi” feature, this will be set automatically.

When generating a book from topics with imagemaps, remember to enable the BuildImagemapHotspots setting in the BookBuildOverrides section of the book-build INI file. If this is not enabled, the hot spots will not be created.

Related information:

"Book-Build INI file" on page 62

Build Map from Outline

Generates a DITA map and stub files from a simple FrameMaker document or text file.

This command builds a DITA map and associated DITA topic files based on the title text in paragraphs in a FrameMaker document (a binary FM file, not an XML file) or a text file opened in FrameMaker. The topic type for each topicref is defined by the paragraph tag name, and the nesting level is defined by the number of tabs that indent each paragraph. You can optionally provide an “element template” to define the initial structure for each new topic file.

A “map from outline” template file is provided in the *Structure\xml\DITA-FMx_1.2* folder. This file provides the paragraph tags for the basic topic types, but you can add your own tags for any specialized topic types that are needed. Any paragraph tags that start with an underscore are ignored when creating topicrefs. The new DITA map file is named based on the FM file’s name and all files are created relative to the FM file.

The paragraph tag named “_map-title” defines the text that will be used for the map/@title attribute (the map’s title). The paragraph tags “topic,” “concept,” “task,” and “reference” are used to define a topicref of the specified type. To specify an element template, include the name of the template after the topic title in angle brackets. For example, in the following paragraphs the first one is tagged with the “concept” style and the second is tagged with the “task” style. When converted into a map, it would create two topics, the first being a concept that used no element template, and the second a task that used the element template named “new~task~basic-task.fm”.

```
Linear Objects
Drawing Lines <basic-task>
```

The file names of the generated topic files are defined by the New File Name Format in the DITA-FMx Options dialog. This may mean that the filenames are title-based or they could be based on the topic’s unique ID, the date/time, or other values. You can also use that format to create the new files in topic-based folders.

If the text of the paragraph is a DITA file name (must end with “.xml”, “.dita”, or “.ditamap” and have no spaces), that file name will be used for the href attribute value. If the paragraph tag is named “Body” it will assume the “topic” type. This allows you to open a text file that is a listing of files, and quickly generate a DITA map from that list.

Related information:

"New DITA File" on page 1
"Creating Element Templates" on page 49
"New File Options" on page 41
"Auto-Prolog Options" on page 44

Build WorkBook from Map

Generates a FrameMaker book that contains all of the XML files referenced by a DITA map.

A “WorkBook” is a FrameMaker book that contains all of the XML files referenced by a DITA map and any sub-maps. This book is used for book-wide processing using FrameMaker’s built-in commands; it is not intended to be used for publishing or output generation. Use the **Open All XML Files in Book** command to open the XML files before using a book-wide processing command.

RELATED INFORMATION:

"Open All XML Files in Book" on page 13

Open All XML Files in Book

Opens all XML files in a “WorkBook” to facilitate the use of book-wide actions.

This command is only available when a FrameMaker book has the focus. In order to take advantage of FrameMaker’s book-wide processing commands (such as spell checking and search), XML files must be opened before running the command (FrameMaker will not automatically open XML files as it does with binary FM files).

This command provides the option to open and resolve references in the XML files or just open the files without resolving references. Opening without resolving references may be preferable under certain circumstances, but you may be warned of invalid spelling errors due to the extra spaces left when an xref or conref doesn’t resolve.

RELATED INFORMATION:

"Build WorkBook from Map" on page 12

Merge Para Tags

Ensures that all paragraph tag names in all currently open documents, exist in all of those documents.

This command is useful for making it easier to set up PDF bookmarks for a PDF created from a book. When creating a PDF from a book, the bookmark list is the collection of all paragraph styles in all documents in that book. The include/exclude settings are defined by the first component in the book. Use the Merge Para Tags command to ensure that all paragraph tags in all files, exist in the first document.

NOTE: This command does not actually "copy" the style definitions, it just ensures that the tag names are the same. This means that it should not modify any existing style definitions.

It is best to run this command on the component templates that are used when building a book from a map, as well as any included files like a title page.

By default, all open files are updated but are not saved, so you can decide which ones to save. If you just want to make sure that the "first" file in the book has all of the styles (usually the title page), you can just save that file and close the others without saving.

RELATED INFORMATION:

"Setting Up PDF Bookmarks" on page 77

Xref to Hyperlink

Builds FrameMaker Hyperlinks at each <xref> or <link> element, to enable live hyperlinks in generated PDF files.

By default, only <fm-xref> or <fm-link> elements become "live" hyperlinks in a PDF generated from FrameMaker (as opposed to a PDF generated through the DITA-OT with the **Generate Output** command). Run the **Xref to Hyperlink** command on an FM file or book to build hyperlinks from each <xref> or <link> element. This command only processes "internal" <xref> and <link> elements (those where the scope attribute is set to something other than "external"). To ensure that external xrefs or links become live hyperlinks, you must enable the **Add Hypertext Marker to External Xrefs** option in the Options dialog before generating the FM file or book.

In order to create proper "gotolink" Hypertext commands, associated Hypertext markers with "newlink" commands are inserted at each element that contains an id attribute.

This command should be run only on generated FM files, not the source XML files. Running this command on XML files results in the Hypertext markers being saved to the XML as processing instructions and needlessly clutters the files.

This can also be done automatically during the Map to Book conversion process by selecting the **Convert Xrefs/Links into Hyperlinks** option in the Book Build Settings dialog found in the DITA Options dialog.

RELATED INFORMATION:

"Setting Up to Use Cross-References" on page 46
"DITA Options" on page 33

Flatten Conrefs

Unlocks all conrefs in the active book or file.

By default, conrefs are wrapped in a locked region similar to a text inset. While working in DITA XML files, this is typically the desired functionality, but once a DITA file has been saved to a FM file, it may be useful for the conrefs to be unlocked. This is a particular issue when the conref contains embedded fm-xrefs, which will not be clickable links in a PDF unless the conref has been unlocked.

This can also be done automatically during the Map to Book conversion process by selecting the **Flatten Conrefs** option in the Book Build Settings dialog found in the DITA Options dialog.

IMPORTANT: *This command can be run on an XML file (or a workbook of XML files), and it will flatten all conrefs. Use caution when running this command; once the conrefs are flattened they cannot be rebuilt.*

RELATED INFORMATION:

"DITA Options" on page 33

Configurator

Simplifies the setup for new installations or configurations.

When you're working in a team environment, it's often a good idea for everyone to use the same settings (or at least start off that way). Rather than providing a list of options to manually enable/disable, use the Configurator command to deploy those settings. In addition to setting specific options defined in the *ditafmx.ini* file, this command can also install your custom structure applica- tions.

To use this feature, create a "configuration INI" file and (optionally) a structure application package (ZIP), then copy those files to a location on your shared network (or local file system). As part of your setup process, you request that each team member run the Configurator command and select the configuration INI file.

This INI file has the following basic structure:

```
[INIUpdates]
Count=N
N=<Section>|<Parameter>|<Value>

[MakerINIUpdates]
Count=N
N=<Section>|<Parameter>|<Value>
```

```
[UserMakerINIUpdates]
Count=N
N=<Section>|<Parameter>|<Value>

[InstallApps]
Count=N
N=<AppName>
```

The INIUpdates section sets the specified values in the user's *ditafmx.ini* file. Each entry in this section specifies the section, parameter, and value for an option. To create this file, you'll need to manually set up your system as needed, then open your *ditafmx.ini* file, and transfer the specific settings into the configuration INI file (which can have any name you like). Each entry in the INIUpdates section is identified with a sequential number (starting from 1). Set the Count parameter to the value of the last entry.

Following is an example of a very simple configuration INI file:

```
[INIUpdates]
Count=4
1=General|AutoSmartSpaces|1
2=General|UseFmDpi|1
3=NewFileOptions|NewTopicFileFormat|<$TITLE_NOSPACELC>
4=NewFileOptions|NewMapFileFormat|_<$TITLE_NOSPACELC>
```

To install structure applications as part of this process, your structure applications must be set up similar to the default DITA-FMx apps. Most importantly, you must be using the "stub" technique for adding the structure application definition.

The MakerINIUpdates and UserMakerINIUpdates sections follow the same logical structure as the INIUpdates section. These two additional sections allow you to modify entries in the *maker.ini* files, both in the "FMHOME" area (in Program Files, with the FrameMaker executable file) and the *maker.ini* in the "user" area (a location like *C:\Users\USERNAME\AppData\Roaming\Adobe\FrameMaker\VERSION*).

NOTE: *This functionality does not allow you to delete an entry or set it to a null value (an empty string).*

The Configurator command uses the InstallApps API to perform the structure application installation from a ZIP file. This requires that you add a special AppInstall section to the *ditafmx.ini* file. This can be done using the INIUpdates section in the configuration INI file. For details on this process, see the InstallApps topic.

Unless the AppInstall section has been added to the *ditafmx.ini* file through other means, you'll need to include those entries in the INIUpdates section of your configuration INI file. To trigger the InstallApps API process, include an InstallApps section in the configuration INI file.

For each structure application, there are two requires entries, “ZIP” and “STUB”. The ZIP entry specifies the location of the ZIP file that contains the app files, and the STUB entry is a relative path to the stub file (structure application definition). An optional “ROOT” entry can define the location where the applications are to be installed on the user’s system. If this is missing, the root location is in the *Structure/XML* folder in the FrameMaker application installation directory. The STUB entry path is relative to this ROOT location. Note that the ZIP entry may specify the same ZIP package for multiple apps that share a common directory structure (like the default DITA-FMx apps).

Following is a configuration INI file that includes the installation of 3 custom structure applications. The ZIP package and configuration INI file are copied to a network location at *Z:\fmx-cfg*, and the apps are installed to the user’s system at *C:\fmapps*:

```
[INIUpdates]
Count=13
1=General|AutoSmartSpaces|1
2=General|UseFmDpi|1
3=NewFileOptions|NewTopicFileFormat|<$TITLE_NOSPACELC>
4=NewFileOptions|NewMapFileFormat|_<$TITLE_NOSPACELC>
5=AppInstall|DITA-MyApp-Topic-1.2-ZIP|Z:\fmx-cfg\DITA-MyApp_1.2
_apps.zip
6=AppInstall|DITA-MyApp-Topic-1.2-STUB|DITA-MyApp_1.2\Topic\str
uctapps-stub_topic_1.2.fm
7=AppInstall|DITA-MyApp-Topic-1.2-ROOT|C:\fmapps
8=AppInstall|DITA-MyApp-Map-1.2-ZIP|Z:\fmx-cfg\DITA-MyApp_1.2_a
pps.zip
9=AppInstall|DITA-MyApp-Map-1.2-STUB|DITA-MyApp_1.2\Map\structa
pps-stub_map_1.2.fm
10=AppInstall|DITA-MyApp-Map-1.2-ROOT|C:\fmapps
11=AppInstall|DITA-MyApp-Book-1.2-ZIP|Z:\fmx-cfg\DITA-MyApp_1.2
_apps.zip
12=AppInstall|DITA-MyApp-Book-1.2-STUB|DITA-MyApp_1.2\Book\stru
ctapps-stub_book_1.2.fm
13=AppInstall|DITA-MyApp-Book-1.2-ROOT|C:\fmapps

[InstallApps]
Count=3
1=DITA-MyApp-Topic-1.2
2=DITA-MyApp-Map-1.2
3=DITA-MyApp-Book-1.2
```

If installing the apps to the FrameMaker application area (no “ROOT” entry specified), the user will need to run FrameMaker “As Administrator” so it can write to the Program Files area.

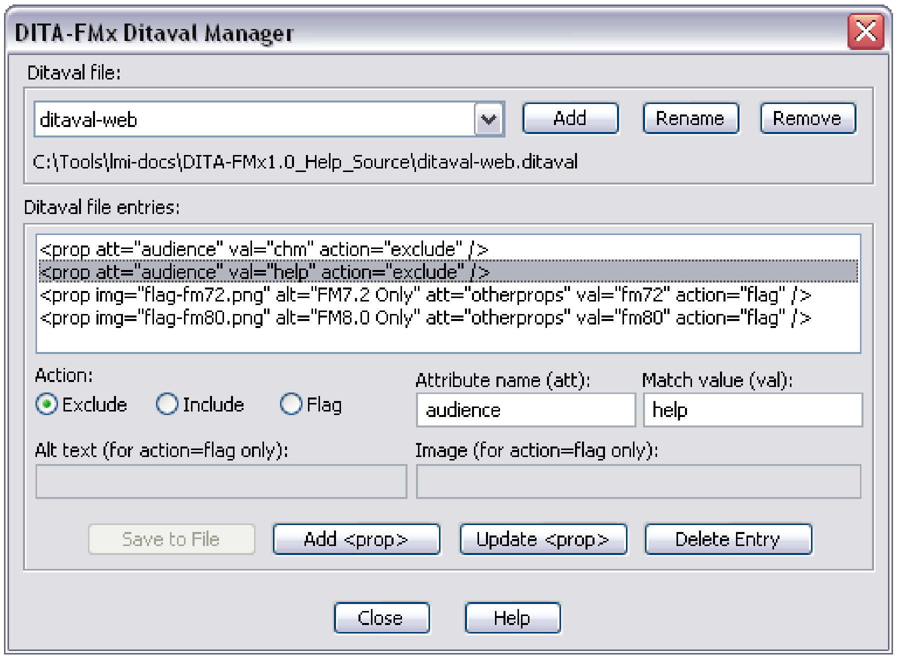
RELATED INFORMATION:

"InstallApps" on page 7
"DITA Options" on page 33
"INI-Only Settings" on page 51

Ditaval Manager

Create and edit ditaval files, and manage the list of ditaval files registered with DITA-FMx.

DITA-FMx provides three commands (**Generate Book from Map**, **Generate Output**, and **Apply Ditaval**) from which you select from a list of ditaval files available to DITA-FMx. The Ditaval Manager controls the files on this list, registered with DITA-FMx. Each ditaval file has an associated name that is displayed in the lists. By default, this name is the ditaval file name, but can be changed if needed.


DITA-FMx Ditaval Manager

Ditaval file: ditaval-web

C:\Tools\lmi-docs\DITA-FMx1.0_Help_Source\ditaval-web.ditaval

Ditaval file entries:

```
<prop att="audience" val="chm" action="exclude" />
<prop att="audience" val="help" action="exclude" />
<prop img="flag-fm72.png" alt="FM7.2 Only" att="otherprops" val="fm72" action="flag" />
<prop img="flag-fm80.png" alt="FM8.0 Only" att="otherprops" val="fm80" action="flag" />
```

Action: [✓ Exclude] Include Flag

Attribute name (att): audience

Match value (val): help

Alt text (for action=flag only):

Image (for action=flag only):

Figure 3-6: DITA-FMx Ditaval Manager dialog box

If you have existing ditaval files that you want to use with DITA-FMx, choose the Add button and select the file. Once it has been added to the list in the Ditaval Manager dialog, this file will be available to the other commands that make use of ditaval files. Use the Rename button to change the name that is shown in the list (this does not change the actual ditaval file name). The Remove button will remove a name from the list and can optionally delete the file as well. You can also use the Add button to create a new ditaval file and add it to the list.

The contents of the currently selected ditaval file displays in the Ditaval File Entries list in the dialog. Selecting a <prop> element makes the attributes of that element available for editing. Only <prop> elements can be edited through this dialog, but all elements (as well as comments) will be shown in the list. After selecting a <prop> element, change the values of the att and val attributes as needed, then choose the Save To File button to write your changes back to the ditaval file.

Use the Add <prop> button to add a new empty prop element to the file, and the Update <prop> button to update the attribute values of the selected entry in the list based on the values in the text boxes. The Delete Entry button deletes the currently selected entry.

The following code shows a simple ditaval file. Each <prop> element has three attributes. The att and val attributes specify the filtering attribute and its value to match on. Valid values for the action attribute are “exclude” and “flag.” This file specifies two filtering schemes, the first excludes all elements whose audience attribute has the value of “admin” and the second excludes all elements whose product attribute has the value of “PROD1.”

```
<?xml version="1.0" encoding="UTF-8"?>
<val>
  <prop att="audience" val="admin" action="exclude"/>
  <prop att="product" val="PROD1" action="exclude"/>
</val>
```

RELATED INFORMATION:

"Apply Ditaval" on page 4
"Generate Output" on page 78
"Setting Up Filtering Groups" on page 32

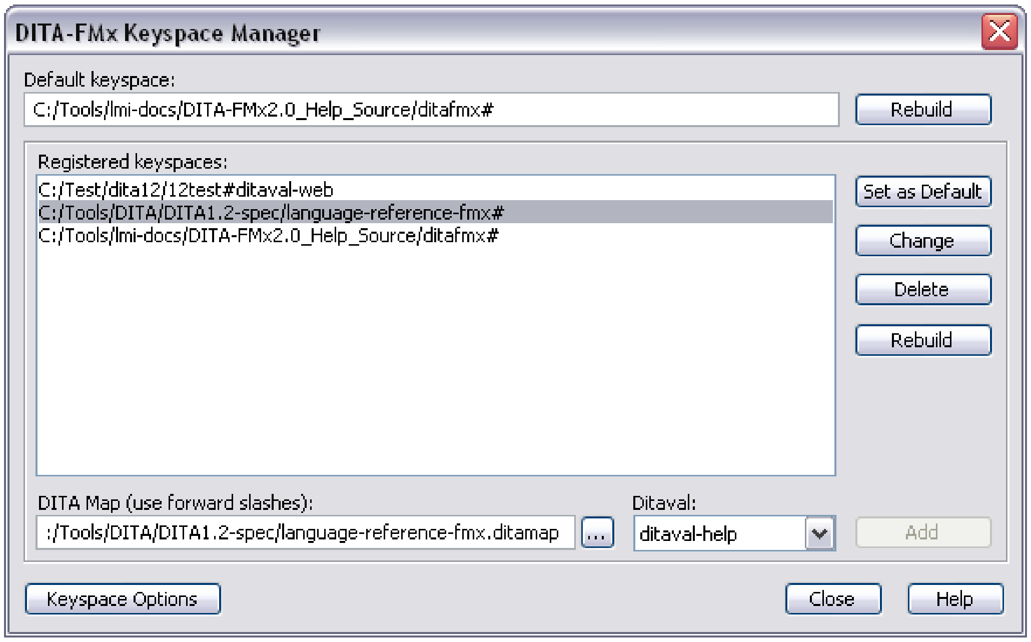
Keyspace Manager

Defines the current and registered keyspaces available to the author.

A keyspace is defined by all key definitions in a map and any submaps, along with any ditaval filtering applied to those key definitions. The **Keyspace Manager** lets you specify the root map for each keyspace and the default ditaval file to be applied to the key definitions found in that scope.

As you insert references to keys, the referenced content or files will be determined by key definitions in the current default keyspace. You should register a keyspace for each root map that contains key definitions or each root map that contains submaps with key definitions. (Do not register a keyspace for submaps unless those maps are used as a root map.)

If you have assigned filtering attributes to your key definitions, and want to author in one specific filtered “view”, assign a ditaval file to the registered keyspace. Each root map can only be registered with one ditaval file at a time. To switch the filtering to render a different ditaval file, change the existing keyspace to use a new ditaval file. The current keyspace filtering has no effect on the filtering applied to the book-build process (which uses the ditaval defined for that build).



C:/Tools/lmi-docs/DITA-FMx2.0_Help_Source/ditafmx#

C:/Test/dita12/12test#ditaval-web
C:/Tools/DITA/DITA1.2-spec/language-reference-fmx#
C:/Tools/lmi-docs/DITA-FMx2.0_Help_Source/ditafmx#

:/Tools/DITA/DITA1.2-spec/language-reference-fmx.ditamap

ditaval-help

Figure 3-7: Keyspace Manager

If keyspaces are registered for multiple maps, when you open one of the registered maps, the **Keyspace Resolution** option (in **Keyspace Options**) can set the default keyspace to match the map. If Keyspace Resolution is set to Auto, this switching is automatic, and if set to Prompt, you are prompted to change the default keyspace.

The **Keyspace Generation** option (in **Keyspace Options**) determines if the default keyspace is regenerated automatically when the associated root map is saved or if you are prompted to regenerate the keyspace (we suggest using "Auto"). If this option is disabled, you can regenerate the keyspace with the **Rebuild** button in the **Keyspace Manager** dialog.

A submap that contains key definitions may be associated with a root map by adding an <othermeta> element with the @name attribute set to "fmx-root-map" and the @content attribute set to the relative path and file name of the root map. When this is done, the submap is treated by the **Keyspace Resolution** and **Keyspace Generation** options as if it were the root map when saving and opening. This ensures that modifications to the key definitions in the submap are reflected in the keyspace.

IMPORTANT: *If the **Keyspace Manager** command is not available, check the "ditaver" attribute for the current file. It must be set to version 1.2 or greater. In order to use keys, you must be using DITA 1.2 or greater. This is determined by the ditaarch:DITAArchVersion attribute on the "topic" element. The DITA* *version is not typically assigned explicitly, but has a default value that will affect the availability of these features in DITA-FMx*

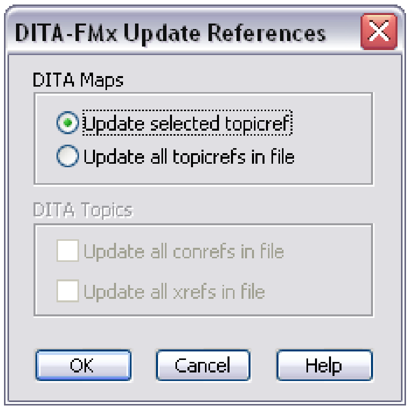
Related information:

"Using Keyspaces in DITA-FMx" on page 54
"Working with Keys" on page 56
"Insert Key Element Reference" on page 27
"Keyspace Options" on page 51
"Map Options" on page 46

Update References

Updates the content of topicrefs, conrefs, xrefs, or links.

Depending on the type of file currently being edited, this dialog provides options for updating references in the file. If a DITA map is active, you have the option to update the selected topicref or all topicrefs in the file. If a DITA topic file is active, you can update xrefs, links, and/or conrefs.



DITA-FMx Update References

DITA Maps
- ● Update selected topicref
- ○ Update all topicrefs in file

DITA Topics
- ☐ Update all conrefs in file
- ☐ Update all xrefs in file

Figure 3-8: DITA-FMx Update References dialog box

References in DITA Maps

Existing topicref elements can be updated so the label text reflects changes in the referenced file's title. To update an existing topicref, select the label and choose the **Update References** command. The **Update References** command also lets you update all topicrefs in the DITA map reflect changes in the referenced files' titles. This command honors the setting of the topicref's locktitle attribute; if locktitle is set to 'yes' the navtitle text is not updated.

Update Selected Topicref

Updates the content of the selected topicref (only if a topicref is selected).

Update All Topicrefs in File

Updates all of the topicrefs in the current file to reflect any changes to titles in referenced files.

References in DITA Topics

Update All Conrefs in File

Updates all of the conrefs in the current file to reflect any changes to the source content.

Update All Xrefs in File

Updates all of the xrefs and links in the current file to reflect any changes to titles in referenced files.

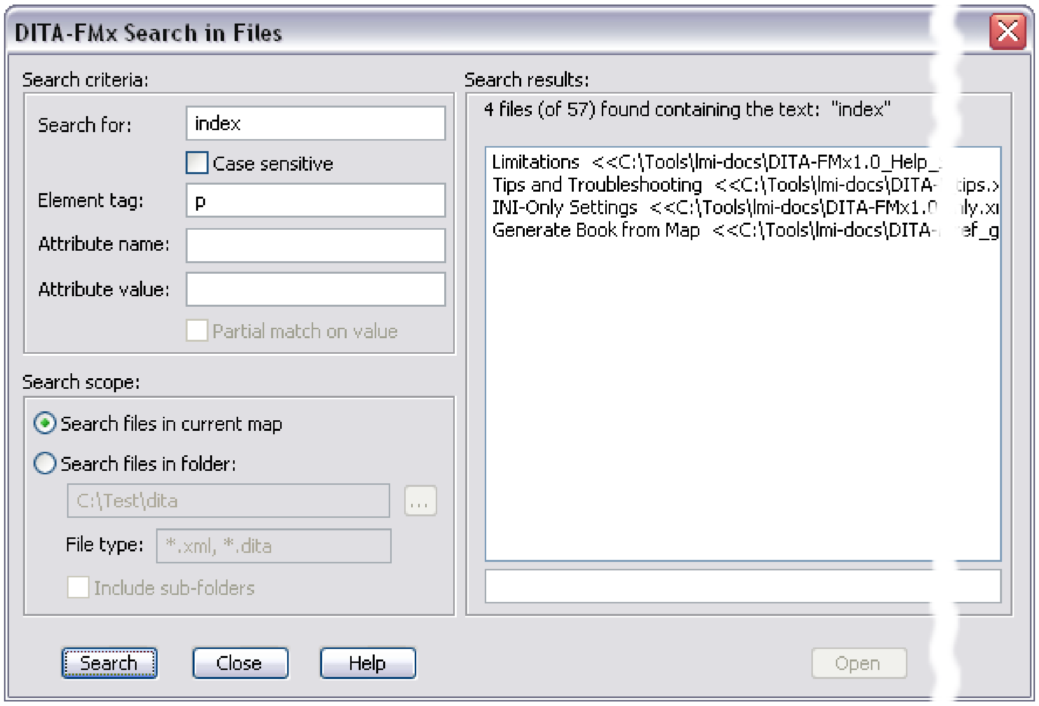
RELATED INFORMATION:

"Using the Reference Manager" on page 18
"Insert Conref" on page 25
"Setting Up to Use Cross-References" on page 46

Search in Files

Search for content in files in a map or on your file system.

Specify search criteria of the following: textual content, element tag name, or attribute name and attribute value. If a DITA map has the focus, you can choose “current map” as the scope, or you can specify a file system path.


DITA-FMx Search in Files

Search criteria:

Search for: index

[ ] Case sensitive

Element tag: p

Attribute name:

Attribute value:

[ ] Partial match on value

Search scope:

(●) Search files in current map

( ) Search files in folder: C:\Test\dita

File type: *.xml, *.dita

[ ] Include sub-folders

Search results:

4 files (of 57) found containing the text: "index"

Limitations <<C:\Tools\lmi-docs\DITA-FMx1.0_Help_
Tips and Troubleshooting <<C:\Tools\lmi-docs\DITA- ips.x
INI-Only Settings <<C:\Tools\lmi-docs\DITA-FMx1.0 ly.xı
Generate Book from Map <<C:\Tools\lmi-docs\DITA- ef_g

Figure 3-9: DITA-FMx Search in Files dialog box

The “Search For” value matches on partial strings (case sensitivity as specified). When searching for an element tag and/or an attribute name, the results will only provide exact matches. Providing an attribute value can optionally match on partial strings, which is useful for locating individual items within a filtering attribute. You can search on the element tag or attribute name only, but if you enter an attribute value, you must also enter the attribute name.

The search results are listed in the dialog showing the topic title and the file name. Select a file and choose Open to open the file.

If the search is taking too long, pressing the Esc key (possibly multiple times) will terminate the search.

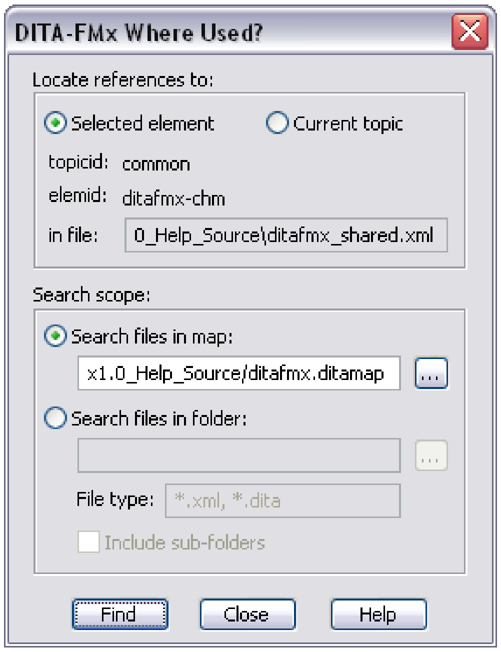
RELATED INFORMATION:

"Where Used" on page 24

Where Used

Generates a report of all references to an element or topic.

Before modifying a topic or referenced element (an element used as a conref), it may be useful to know all of the files that reference that topic or element.



topicid: common
elemid: ditafmx-chm
in file: 0_Help_Source\ditafmx_shared.xml
Search files in map: x1.0_Help_Source/ditafmx.ditamap
File type: *.xml, *.dita

Figure 3-10: DITA-FMx Where Used dialog box

In the Where Used dialog, select the option to indicate the type of reference to locate (element or topic). If the insertion point is in an element that has an ID, both options are available, otherwise only the topic option can be used. Specify the scope as a DITA map or a folder. If you specify a map, the report will be generated based on all files referenced by that map (and any submaps). If you specify a folder, the report will be generated based on all files of the type specified in that file system path.

NOTE: In order to use the Where Used command to work properly for a selected element, the element with the ID value must be selected. This may be obvious in most cases, but if your conref source contains a child element, you need to make sure the insertion point is in the parent element not the child.

The context menu (right-click) includes a Where Used menu item for quick access.

RELATED INFORMATION:

"Insert Conref" on page 25

Insert Topicrefs

Inserts one or more topicrefs at the current insertion point in a map.

NOTE: This command is only available on FM10 and later.

Use this command to insert multiple topicrefs at the current insertion point in a DITA map. A file selection dialog displays where you can choose the topics to insert using SHIFT+CLICK. Due to limitations in the FM API, when selecting multiple files they must be sequential. This command will ignore selected “backup” and “lck” files.

The new elements are inserted as <topicref> elements. If you need different topic referencing elements, change them after insertion.

Related information:

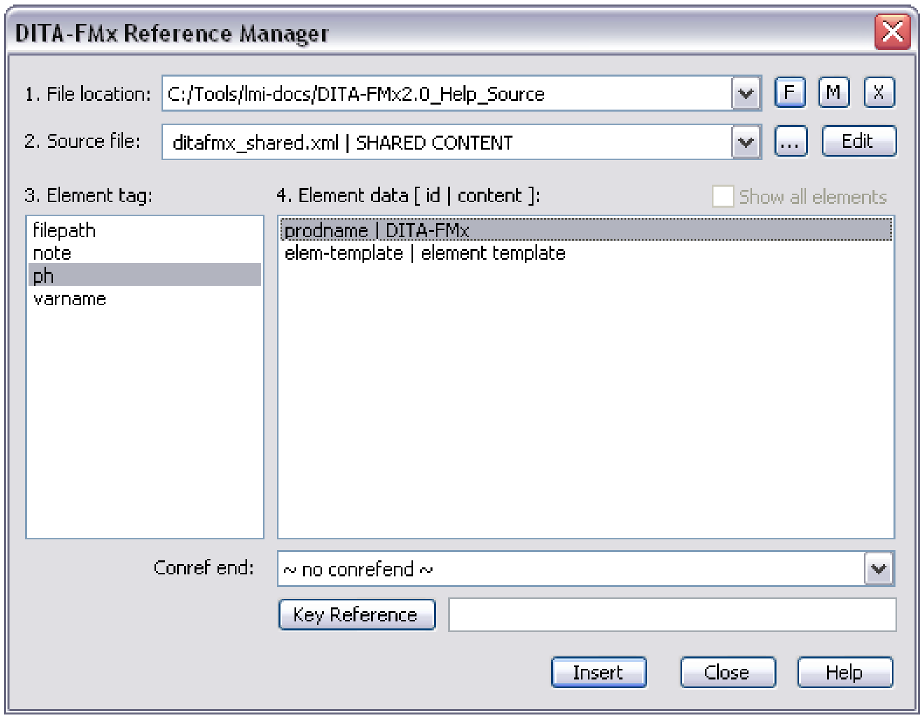
"Working with Maps" on page 49

Insert Conref

Displays the Reference Manager which lets you insert a content reference.

The **Reference Manager** (displayed when you choose **DITA-FMx > Insert Conref**) allows you to create a content reference (conref or conkeyref) to elements within the same file or in other files.

Select a source file from those currently open or from files in a folder or map (as defined by the **File Location** option). Select the target element type in the **Element Tag** list, then select an item from the **Element Data** list to create a conref to that element. Alternatively, you can use the **Key Reference** button to select the target content based on a key.



1. File location: C:/Tools/lmi-docs/DITA-FMx2.0_Help_Source
2. Source file: ditafmx_shared.xml | SHARED CONTENT
3. Element tag: filepath, note, ph, varname
4. Element data [id | content]: prodname | DITA-FMx; elem-template | element template
Conref end: ~ no conrefend ~

Figure 3-11: DITA-FMx Reference Manager dialog box

The **Element Tag** list displays the names of elements that exist in the source file and have an @id attribute value. The **Element Data** list displays the @id and text of the available elements.

If you are creating a conref range, select the end element from the **Conref End** list. Only elements that follow the selected item in the **Element Data** list are available as a conref end element.

To create a reference to an element defined by a key (conkeyref), choose the **Key Reference** button. The **Keyref Manager** dialog lets you select a key from the specified keyspace, then locate the required element id. For a conref, you should select both the key and the element id. Choosing **OK** in the **Keyref Manager** inserts the <key>/<elemid> into the **Key Reference** field.

The referenced element is inserted at the location specified. You can double click the conref to re-open the Reference Manager to change the referenced element or edit the source file. The Reference Manager also lets you display the list of all elements for conrefing (even if they have no @id value), you must provide the @id value at insertion time.

On the opening of a file, the content of any element included by conref is resolved and displayed as a locked text range (similar to a text inset). The color of conrefs is defined by the “DITA-Conref” color (this is a custom color definition and can be changed in the template). The auto-loading functionality may be enabled/disabled with the **Options** command.

The **Update References** command provides an option to load and build the conref elements (if they were not initially loaded by the auto-load functionality), and update the conrefs to reflect changes in the source files.

If you want to “break” a conref (make it into editable text rather than a locked range), select the conref and choose the **Flatten Conref** command (**DITA-FMx > FM File Commands > Flatten Conrefs**). To flatten all conrefs in the current file, use the same command, but place the insertion point away from a conref.

RELATED INFORMATION:

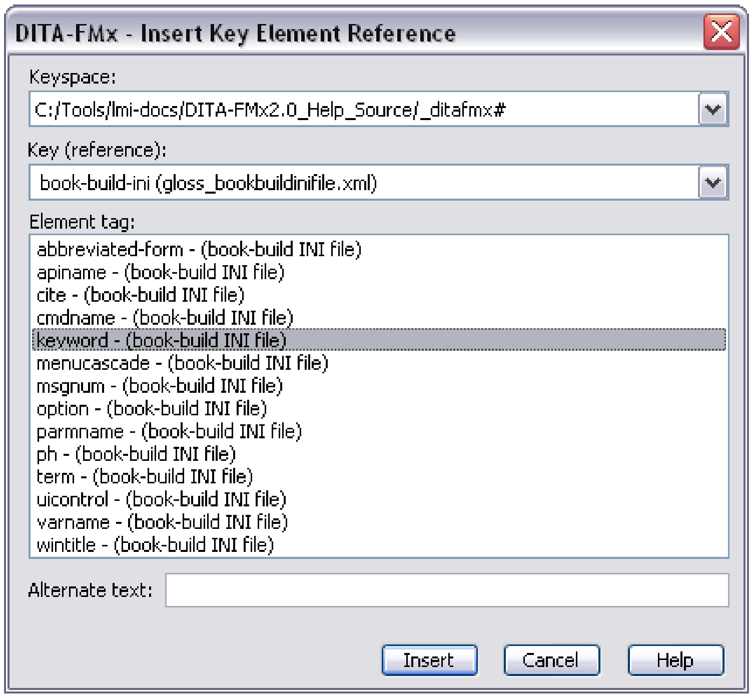
"Using the Reference Manager" on page 18
"Update References" on page 21
"Using coderefs" on page 61

Insert Key Element Reference

Inserts an element whose content is defined by content within a key definition.

NOTE: *The term “key element reference” is not an industry standard term. We use this term in lieu of any other useful term for this DITA construct.*

The **Insert Key Element Reference** dialog lets you select the keyspace and the key, then it displays the available elements and their current values.



Keyspace: C:/Tools/lmi-docs/DITA-FMx2.0_Help_Source/_ditafmx#

Key (reference): book-build-ini (gloss_bookbuildinifile.xml)

Element tag:

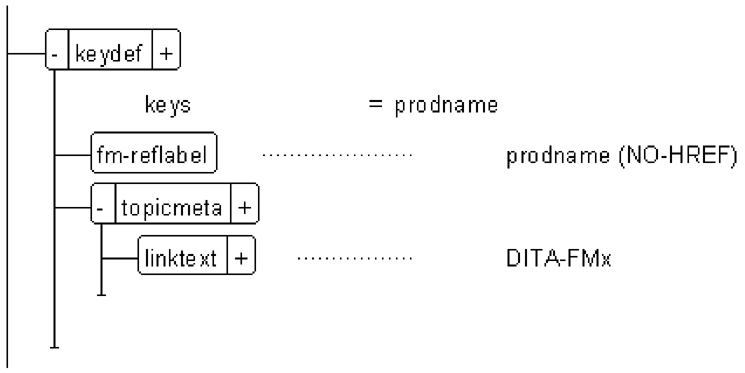
```
abbreviated-form - (book-build INI file)
apiname - (book-build INI file)
cite - (book-build INI file)
cmdname - (book-build INI file)
keyword - (book-build INI file)
menucascade - (book-build INI file)
msgnum - (book-build INI file)
option - (book-build INI file)
parmname - (book-build INI file)
ph - (book-build INI file)
term - (book-build INI file)
uicontrol - (book-build INI file)
varname - (book-build INI file)
wintitle - (book-build INI file)
```

Alternate text:

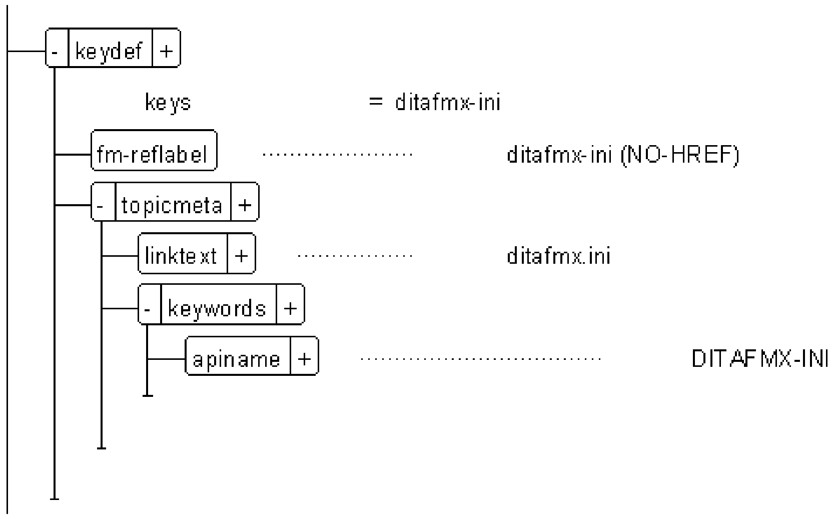
Figure 3-12: Insert Key Element Reference dialog

A <keydef> that contains a <topicmeta> element with a <linktext> child will add the content of <linktext> into any element that you insert which references the associated key.

If a <keydef> does not define any a <topicmeta> content, the title text from the referenced topic is used as the content for the key element reference. You can override the title text by providing an Alternate Text value.



To define the content of specific key-based elements, include those elements in the <topicmeta> element. In the following example key definition of a “ditafmx-ini” key, inserting the <apiname> element will provide the content “DITAFMX-INI” for references that use the <apiname> element while all other elements will use the content of the <linktext> element as “ditafmx.ini.”



IMPORTANT: *While the use of the <linktext> element as described above is clearly indicated as valid in the DITA specification, processing by the DITA Open Toolkit does not seem to honor this structure. You may want to use the alternate method described below, which does work in OT processing.*

If your <keydef> includes a <keyword> element (within <keywords>), that value will also be used when inserting a key element reference of non-matching element types. For example, if you insert a key-referencing <ph> element, the content within the <ph> will be that of the <keyword> element.

References to glossary entries

DITA-FMx supports the use of key-based references to glossary entries. When setting up a map that contains topicrefs to <glossentry> topics, assign a key to that <topicref> (or <glossref>) so the glossary term can be referenced by a key. (If using a <topicref>, remember to set the @type attribute to "glossentry".)

Assuming that your glossary entries are identified by a key, insert a <term> element with the @keyref set to the value of the key. While authoring, the content of the <term> will match that of the glossary entry's title (the <glossterm> element).

If you enable the **Glossary Term Swapping** option (in the **Book Build Settings** dialog), the <term> will render the <glossSurfaceForm> content in the first instance of the <term> in a book component, and the value of <glossAlt> (<glossAcronym> or <glossAbbreviation>) for remaining instances.

Related information:

"Using Keyspaces in DITA-FMx" on page 54
"Working with Keys" on page 56
"Keyspace Manager" on page 20
"Keyspace Options" on page 51
"Map Options" on page 46

Assign ID to Element

Assigns a generated ID to the selected element.

The type of generated ID is based on the option selected in the DITA Options dialog. The ID type "GUID" is a standard globally unique identifier, and the ID type "QUID" is a quasi-unique identifier.

The QUID value is based on the current date and time (composed from values representing the year, month, day, hours, minutes, seconds, plus two randomly generated values). It is designed to be unique per user for 100 years. You can specify an ID prefix in the Options dialog. If you specify a unique ID prefix for each user, the generated IDs will be unique for each member of your team.

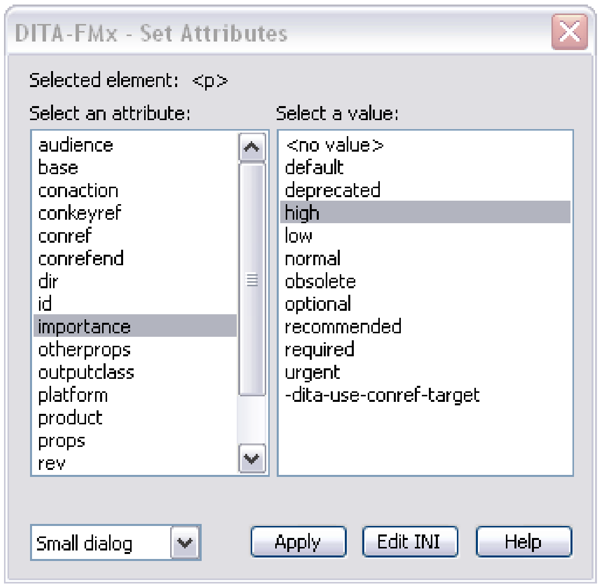
Related information:

"DITA Options" on page 33

Set Attributes

Provides a quick and easy way to set attribute values and provide custom project-specific attribute values.

The Set Attributes command displays a modeless dialog that lets you easily set the attribute values on selected elements. Attribute names are displayed in a listbox on the left of the dialog, and when one is selected, an appropriate form field is displayed on the right for you to enter or select the value to apply. Use the popup list box in the lower left of the dialog to set the size of the dialog. The larger dialogs allow selection of more values.


DITA-FMx - Set Attributes

Selected element: <p>

Select an attribute:
- audience
- base
- conaction
- conkeyref
- conref
- conrefend
- dir
- id
- importance
- otherprops
- outputclass
- platform
- product
- props
- rev

Select a value:
- <no value>
- default
- deprecated
- high
- low
- normal
- obsolete
- optional
- recommended
- required
- urgent
- -dita-use-conref-target

Figure 3-13: DITA-FMx Set Attributes dialog box, single value selection

Depending on the underlying attribute type (Choice, String, Strings, Integer, Integers, Real, Reals, IDReference, IDReferences, and UniqueID), as defined in the EDD, the attribute values display in a different type of field. For a Choice attribute, the values specified in the EDD by the Choices element are displayed in a scrolling list box. For the Strings, Integers, Reals, and IDReferences attribute types, the values specified in the EDD by the "Default" elements is displayed as an array of checkboxes (up to 10 in the small dialog, 20 in the medium dialog, and 40 in the large dialog), allowing you to select one or more values. All other attribute types are displayed in a simple text box. The checkboxes are particularly useful for managing the DITA filtering attributes (platform, product, audience, and otherprops).

Because the dialog is modeless, you can leave it open and whenever desired, select an element and apply new attribute values. In addition to the main DITA-FMx menu, the Set Attributes command is available from the context (right-click) menus.

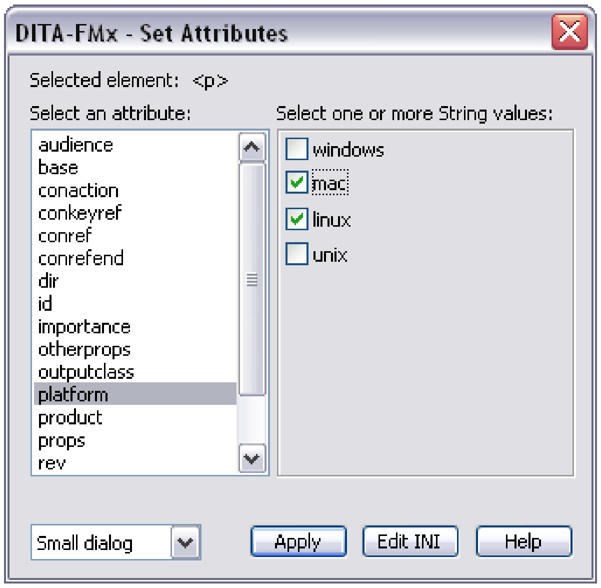
The Set Attributes command provides a feature that lets you extend the values displayed for specific attributes, by allowing these to be specified in a text file. It also lets you specify different predefined attribute values for different projects. This mechanism lets you associate a *filtering group* name with a file system path. When editing a DITA file that is within the specified path, the values associated with that group are displayed with those defined in the EDD. These groups and their associated attribute names and possible values are defined in an INI file named (by default) *FilterGroups.ini* that is created in the user's DITA-FMx folder (**DITA-FMx > Open DITA-FMx Folder**). You can edit this file by choosing the **Edit INI** button in the Set Attributes dialog.

The following sample filtering groups file defines two groups named ProductA and ProductB (these are both defined as an attribute of type "Strings" in the EDD). When you edit a DITA file that is in the path specified by group ProductA, selecting the product attribute will offer the three products listed (in addition to any already defined in the EDD) as options. Likewise, selecting audience or platform will offer those items as options.

```
[General]
ProductA=C:\projects\product-a
ProductB=C:\projects\product-b

[ProductA]
product=ProdALite|ProdAFull|ProdASimple
audience=Novice|Expert|Admin
platform=windows|mac|linux|unix
outputclass=style1|style2|style3|style4|style5|style6

[ProductB]
product=ProdBLite|ProdBFull
audience=User|Developer|Admin
platform=windows|mac|linux|unix
outputclass=style1|style2|style3|style4|style5|style6
```



Selected element: <p>

Select an attribute: platform

Select one or more String values:

- [ ] windows
- [✓] mac
- [✓] linux
- [ ] unix

Figure 3-14: DITA-FMx Set Attributes dialog box, multiple value selection

This mechanism can also be used for attributes defined as a type of "String" to predefine mutually exclusive attribute values similar to the attribute type of "Choice." In the previous example, selecting the outputclass attribute (defined as a "String" in the EDD) from the list, would display a scrolling list box with the style names available for selection.

There are three "INI-Only" parameters that can be used to enhance the functionality of this command.

SetAttrStrings

Specifies an alternate file to use as the filtering groups INI file. Useful to define a shared file at a network location.

SetAttrIgnore

Specifies the attributes to ignore (and not display) in the Set Attributes dialog.

SetAttrStringsDefault

Specifies the "default" value that is ignored when displaying the list of defaults.

To modify these settings, you must manually edit the *ditafmx.ini* file and update (or add) these values to the INIOnly section. For more information see INI-Only Settings.

Related Information:

"INI-Only Settings" on page 51
"Ditaval Manager" on page 18
"Setting Up Filtering Groups" on page 32

Setting Up Filtering Groups

PREREQUISITES:

Before you can make use of filtering groups, your EDD must be set up to use the "Strings" attribute type for the attributes that you want to use with a filtering group. You can optionally add "default" values to the attribute definitions in the EDD, but that is not required (and may not be desirable). At a minimum, just change the "String" type to "Strings" and you should be all set.

The default DITA-FMx Topic and Map templates use the Strings type for all instances of the platform, product, audience, and otherprops attributes.

The following task creates two filtering groups named "ProductA" and "ProductB" and sets up unique values for each group that are available to apply to the attributes specified. Feel free to change the names and file paths to match those on your system.

TASK

1. Using a text editor such as Notepad, create a file named *filtergroups.ini* in the user's DITA-FMx folder (**DITA-FMx > Open DITA-FMx Folder**).
2. Create a "General" section that defines the group names and the associated root paths, then create a section for each group that defines the attribute names and available values.

```
[General]
ProductA=C:\projects\product-a
ProductB=C:\projects\product-b

[ProductA]
product=ProdALite|ProdAFull|ProdASimple
audience=Novice|Expert|Admin
platform=Windows|Mac|Linux
otherprops=001|002|003|004|005|006

[ProductB]
product=ProdBLite|ProdBFull
audience=User|Developer|Admin
platform=Windows|Mac|Linux|Solaris
```

3. Save this file.

AFTER COMPLETING THIS TASK:

In FrameMaker, when you open a file in Product A (somewhere below the path *C:\projects\product-a*), when you run the Set Attributes command, and select one of the filtering attributes from the list, you will be able to select one or more of the values specified in the INI file.

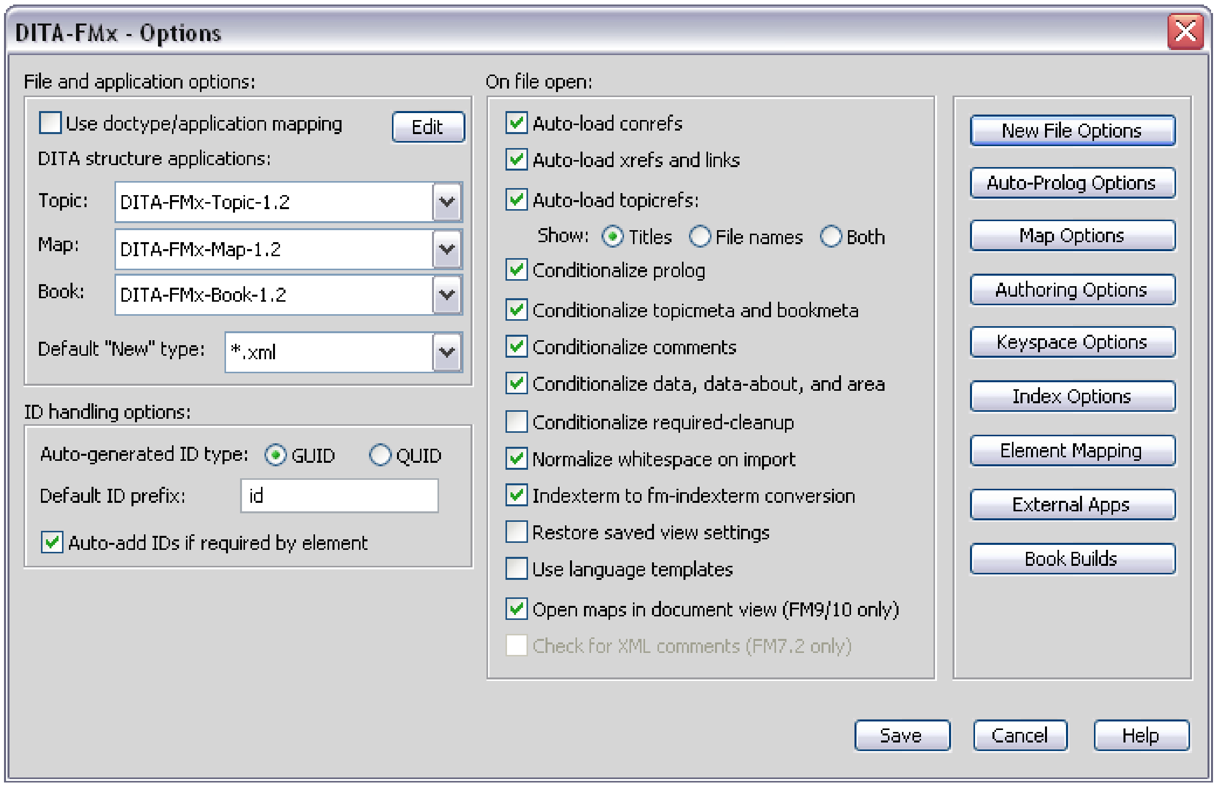
Related information:

"Set Attributes" on page 30

DITA Options

Specify options that control the various authoring and publishing features of DITA-FMx.

The Options dialog provides access to the frequently modified properties and settings. Other settings can be changed manually in the *ditafmx.ini* file in the user's DITA-FMx folder (**DITA-FMx > Open DITA-FMx Folder)**, see the INI-Only Settings topic for details.


DITA-FMx - Options

File and application options:

- [ ] Use doctype/application mapping — Edit
- DITA structure applications:
  - Topic: DITA-FMx-Topic-1.2
  - Map: DITA-FMx-Map-1.2
  - Book: DITA-FMx-Book-1.2
- Default "New" type: *.xml

ID handling options:

- Auto-generated ID type: [✓ GUID] QUID
- Default ID prefix: id
- [x] Auto-add IDs if required by element

On file open:

- [x] Auto-load conrefs
- [x] Auto-load xrefs and links
- [x] Auto-load topicrefs:
  - Show: [✓ Titles] File names Both
- [x] Conditionalize prolog
- [x] Conditionalize topicmeta and bookmeta
- [x] Conditionalize comments
- [x] Conditionalize data, data-about, and area
- [ ] Conditionalize required-cleanup
- [x] Normalize whitespace on import
- [x] Indexterm to fm-indexterm conversion
- [ ] Restore saved view settings
- [ ] Use language templates
- [x] Open maps in document view (FM9/10 only)
- [ ] Check for XML comments (FM7.2 only)

Figure 3-15: DITA-FMx Options dialog box

File and Application Options

Use Doctype/Application Mapping

Uses a doctype-to-structure-application mapping to specify the application to use when opening a DITA topic or map file. Choose the Edit button to define the mapping, then enable the **Use doctype/application mapping** option to use the mapping. For details on this feature, see Doctype/Application Mapping.

It is important to define a mapping for all possible doctypes that you may encounter, otherwise errors will result when opening files. When this option is enabled, the Topic and Map applications specified in the Options dialog are ignored. The “doctype” is the root element’s name, or the “topic type”. If you use the task, concept, and reference topic types, you should set up a mapping for each type.

Use of this option allows you to define separate structure applications for each topic type, or you can use a structure application that supports multiple types (like the default DITA-FMx applications). The structure applications can share common files as needed (typically the EDD and DTDs), but each unique application name will require a separate entry in the structure application definitions file.

DITA Topic Application

The name of the Topic structure application used for topic items on the **New DITA File** menu. The Topic structure application is used for authoring of DITA topics.

Also, if the **Use doctype/application mapping** option is not enabled, this specifies the application used to open a DITA topic file when that topic is opened automatically (such as when you double-click a topicref or reference a topic from a topic or a map).

DITA Map Application

The name of the Map structure application used for map items on the **New DITA File** menu. The Map structure application is used for authoring of DITA maps.

Also, if the **Use doctype/application mapping** option is not enabled, this specifies the application used to open a DITA map file when that map is opened automatically (such as when you double-click a topicref or reference a map from another map).

DITA Book Application

The name of the Book structure application used by the **Generate Book from Map** command. The Book application is used for generating FM book and chapter files from a DITA map; it is not used for authoring.

Default "New" Type

Specifies the file extension applied to new topic files when one is not provided as part of the file name.

ID Handling Options

Auto-Generated ID Type

Specifies the type of ID that will be generated automatically by DITA-FMx when an ID is required. GUID specifies a globally unique ID and QUID specifies a "quasi unique" ID. The QUID is shorter and when combined with a unique ID prefix can be considered to be unique under typical conditions (but is technically not globally unique).

Neither of these ID types is necessarily better than the other, it really depends on your needs. The GUID is defined to be "globally" unique, so if this is important to your process that might be a reason to use the GUID. The downside of a GUID is that they are very long and not particularly user-friendly. The QUID is unique under most conditions, and if each of your writers specifies a unique prefix the uniqueness is almost guaranteed. The QUID is probably preferable if you want to be able to "read" the values.

Default ID Prefix

The string that is used as a prefix on IDs that are automatically generated.

Auto-Add IDs if Required by Element

When an element is inserted that has a required ID attribute, that attribute value is automatically added.

On File Open

Auto-Load Conrefs

On file open, any conrefs are resolved and updated. Note that this auto-loading is applied to all files opened as a result of a reference in that file being resolved. The number of levels of reference resolution is determined by the MaxRefLevels INIOnly parameter.

Auto-Load Xrefs and Links

On file open, any <xref> (or <fm-xref>) and <link> (or <fm-link>) elements are resolved and the labels are updated with the text of the target element. Note that this auto-loading is applied to all files opened as a result of a reference in that file being resolved. The number of levels of reference resolution is determined by the MaxRefLevels INIOnly parameter.

If this option is disabled, <fm-xref> and <fm-link> elements are not converted from their base <xref> or <link> elements; they will remain as <xref> or <link> elements until you enable this option.

Auto-Load Topicrefs

On file open of a DITA map, labels (as <fm-reflabel> and <navtitle> elements) are added to all topicref-based elements. To open the associated file, double-click the label. If this option is selected, the additional "Show" options are available:

- **Titles** - displays the target file's title as the label.
- **File names** - displays the target file's file name as the label (the value of the href attribute).
- **Both** - displays the target file's title and file name as the label.

Conditionalize Prolog

On file open, the <prolog> element is tagged with the "DITA-Prolog" condition. If this condition does not exist, it is created, and set to "Show." If this condition already exists in the template, the condition is applied and the current Show/Hide state is used.

Conditionalize Topicmeta and Bookmeta

On file open (of a map), any <topicmeta> and <bookmeta> elements are tagged with the "DITA-Topicmeta" condition. If this condition does not exist, it is created, and set to "Show." If this condition already exists in the template, the condition is applied and the current Show/Hide state is used.

Conditionalize Comments

On file open, any <draft-comment> elements are tagged with the "DITA-Comment" condition. If this condition does not exist, it is created, and set to "Show." If this condition already exists in the template, the condition is applied and the current Show/Hide state is used.

Conditionalize Data, Data-About, and Area

On file open, any <data>, <data-about>, and <area> elements are tagged with the "DITA-Data" condition. If this condition does not exist, it is created, and set to "Show." If this condition already exists in the template, the condition is applied and the current Show/Hide state is used.

NOTE: *This option applies the DITA-Data condition to all <data>-based elements, such as <glossPartOfSpeech>, <glossStatus>, <glossProperty>, and others.*

Conditionalize Required-Cleanup

On file open, the <required-cleanup> elements are tagged with the "DITA-Cleanup" condition. If this condition does not exist, it is created, and set to "Show." If this condition already exists in the template, the condition is applied and the current Show/Hide state is used.

Normalize Whitespace on Import

Strips redundant spaces and tabs from the XML file, often added by XML editors to "pretty-print" XML files for ease of use. If this option is disabled, you may see extra whitespace or paragraphs in the file when open in FrameMaker.

Indexterm to Fm-Indexterm Conversion

If enabled, <indexterm> elements are converted into <fm-indexterm> elements on file open and that process is revered when the file is written to disk. This provides for compatibility between the DITA indexing syntax and FrameMaker's marker syntax. If this option is not enabled, <index-term> elements import as container elements and display inline as ordinary content.

IMPORTANT: If an <indexterm> element contains child elements (other than <index-see>, <index-see-also>, and <index-sort-as>), those elements will be lost on import. If your files do contain additional child elements within an <indexterm>, you should disable this option.

Restore Saved View Settings

On file open, the "saved" view settings are restored. The settings are saved when the **Save View Settings** command is run, and include the document zoom value, attribute display options, and the visibility of borders, text symbols, rulers, grid lines, and element boundaries.

Use Language Templates

Enables the use of language-specific template files. When opening a DITA XML file, if the topic's @xml:lang attribute specifies a value, and a matching language template exists in the same folder as the default structure application template, that template is used instead of the default template. The language template must be named "<templatename>.<langval>.fm" (the ".fm" extension is optional, but if the default template includes the extension, the language template must also. For example, if your default template is *topic.template.fm*, a Japanese language template would be named *topic.template.ja-jp.fm* (assuming that "ja-jp" was the value in the xml:lang attribute).

If this option is enabled, DITA-FMx will also look for a language-specific book-build INI file when creating a book from a map (*ditafmx-book-build.<langval>.ini*). This file will be located in the same places as the standard book-build INI file. If a language-specific INI is not found, the default book-build INI will be used (if available). For example, if the @xml:lang attribute value is set to "ja-jp", DITA-FMx will look for the file *ditafmx-bookbuild.ja-jp.ini*.

NOTE: In DITA-FMx 2.0.01, this setting was migrated from an "INI-Only" UseLanguageTemplate setting into the Options dialog.

Open Maps in Document View (FM9/10)

When opening a DITA map, instead of opening it in the **Resource Manager**, it is opened in the document view. Although the **Resource Manager** presents the files in an orderly package, it is not a terribly useful way to work on a map because many features of a map are unavailable.

NOTE: On FM9/10 this option operates properly on initial map open and when opening a map from a topicref. For later versions, it only works when opening a map from a topicref; after initial map open, you'll need to manually switch to document view.

Check for XML Comments (FM7.2)

On file open, a message displays at the console window if the file contains XML comments. This option is not available (or needed) in FM versions above FM7.2 since XML comments now round-trip as markers.

Additional Options

New File Options

Displays the New File Options dialog where you can specify the new file name format for topics and maps, as well as the location of the element template folder. This dialog also controls the default setting for wrapping new topics in the dita element.

Auto-Prolog Options

Displays the Auto-Prolog Options dialog which lets you control the way the <prolog>, <topicmeta>, and <bookmeta> are initialized for new files and updated on file open.

Map Options

Displays the Map Options dialog which controls the way the <navtitle> element is rendered when a map is opened and how the <navtitle> is added when a topicref-based element is created.

Authoring Options

Displays the Authoring Options dialog which provides numerous settings that affect authoring.

Keyspace Options

Displays the Keyspace Options dialog which controls the way a keyspace is generated and resolved.

Index Options

Displays the Index Options dialog which lets you control the rendering and functionality of <index-see> and <index-see-also> elements.

Element Mapping

Displays the New Element Mapping dialog. This dialog lets you control the rendering of <simpletable>-based elements and preformatted elements.

External Apps

Displays the External Application Settings dialog. This dialog specifies the following program locations and settings:

- DITA Open Toolkit installation directory (used by the **Generate Output** command)
- utility used for the **Create Archive** command
- application used as the text editor for coderefs

Book Builds

Displays the Book Build Settings dialog. This dialog provides control over the processes that are run on the book and generated FM files created by the **Generate Book from Map** command.

Related information:

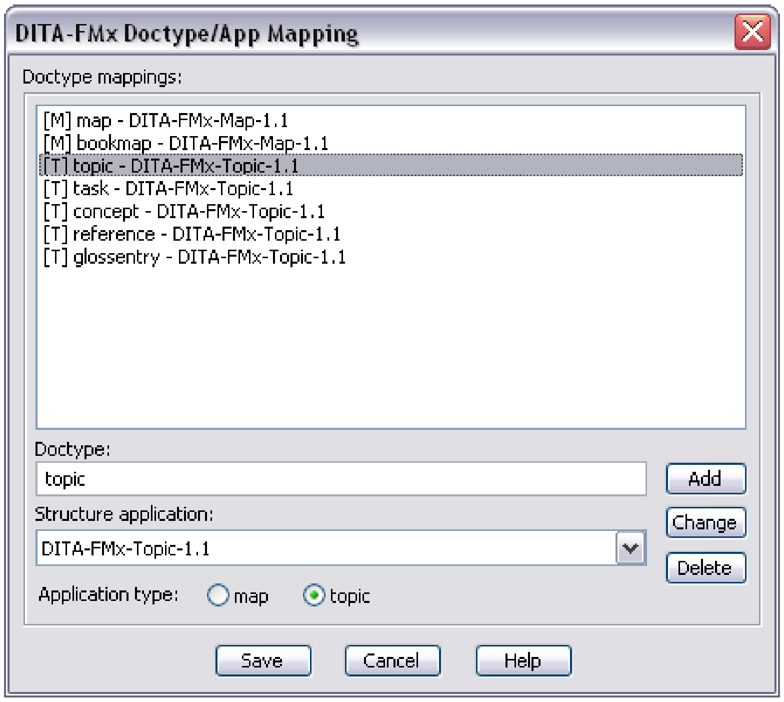
"INI-Only Settings" on page 51
"Developing Custom Structure Applications" on page 21
"Save View Settings" on page 9

Doctype/Application Mapping

Associate individual structure applications with each topic type.

The **Doctype/Application Mapping** dialog lets you associate individual structured applications with each topic type (or doctype). The default DITA-FMx structured application provides support for all of the main DITA topic types in a single application. This simplifies template and EDD maintenance, but does mean that all of the topics use the same doctype, ditabase.

If you want to maintain the applications separately or need to use the topic-specific doctypes, you should use the doctype/application mapping feature.



```
[M] map - DITA-FMx-Map-1.1
[M] bookmap - DITA-FMx-Map-1.1
[T] topic - DITA-FMx-Topic-1.1
[T] task - DITA-FMx-Topic-1.1
[T] concept - DITA-FMx-Topic-1.1
[T] reference - DITA-FMx-Topic-1.1
[T] glossentry - DITA-FMx-Topic-1.1
```

Doctype: topic

Structure application: DITA-FMx-Topic-1.1

Application type: map [✓ topic]

Figure 3-16: Doctype/App Mapping dialog box

The mapped doctypes (or topic types) are shown in the scrolling list box. The application type is indicated by a letter in square brackets, an “M” for maps or a “T” for topics, followed by the doctype name, then the structured application name.

Add

To add a doctype mapping, enter the doctype and select the structure application from the list. Select the application type (map or topic) and choose the Add button. The new mapping is added to the bottom of the list.

Change

To modify an existing doctype mapping, select the mapping from the list. The doctype, structure application, and application type values populate the fields in the bottom of the dialog box. Modify those fields as needed, then choose the Change button.

Delete

To delete a doctype mapping, select the mapping from the list then choose the Delete button.

When you are done making changes to the mappings, choose the Save button.

FM 10-12 If you are using FM10 or later, the mapping you define will be added to the default FrameMaker ditafm.ini file in a new section named

"DITA-FMx_1.1_Applications". Also, by default, the DITA version will be set to "DITA 1.1" (even for DITA 1.2 apps).

Also, on FM10 and later, you may need to restart FrameMaker after changing structure applications to ensure that the proper structure applications have been registered in the *ditafm.ini* file. Due to changes in these FrameMaker versions, the structure applications are stored in both the *ditafmx.ini* and the default *ditafm.ini* files.

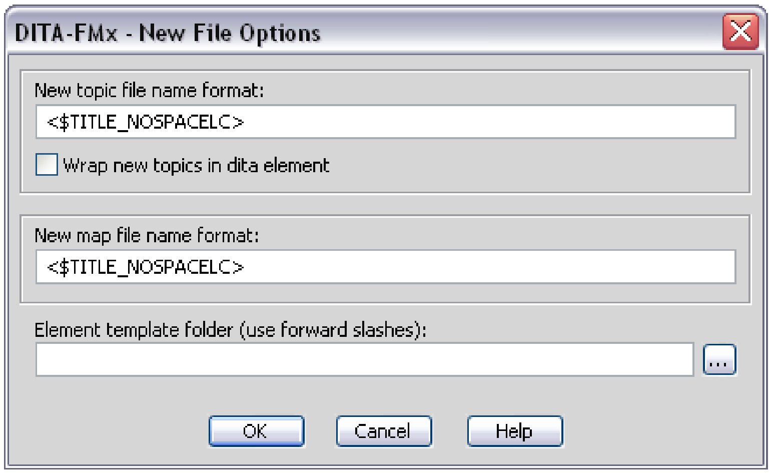
RELATED INFORMATION:

"DITA Structure Applications" on page 11

New File Options

Provides settings that affect the creation of new topic and map files.

The new file options affect the functionality of the **New DITA File** dialog. You can access these options from the main **Options** dialog as well as the **New DITA File** dialog.


DITA-FMx - New File Options

New topic file name format: <$TITLE_NOSPACELC>

Wrap new topics in dita element: ☐

New map file name format: <$TITLE_NOSPACELC>

Element template folder (use forward slashes):

Figure 3-17: DITA-FMx New File Options dialog box

New Topic File Name Format

This field defines the text of auto-generated file names for topics. File names are auto-generated in the **New DITA File** dialog as well as the **Build Map from Outline** command. You can enter plain text in this field as well as special building blocks. Details on use of building blocks is provided below.

Wrap New Topics in DITA Element

When a new topic file is created (using the **New DITA File** command), the new file is created with a <dita> element at its root. If you plan to include multiple topics in a single file, that file must have <dita> as the root element. This option controls the default setting for this feature, it can be overridden by the same option in the New DITA File dialog.

New Map File Name Format

This field defines the text of auto-generated file names for maps. File names are auto-generated in the **New DITA File** dialog as well as the **Build Map from Outline** command. You can enter plain text in this field as well as special building blocks. Details on use of building blocks is provided below.

Element Template folder

Specifies the folder where element templates are stored. This can be a local or network location. If the element template folder field is empty, the default location is the folder that contains the structure application template file.

NOTE: When entering paths in text fields, you must use the forward slash as the directory separator.

Use of building blocks

A building block is a string of text enclosed in angle brackets. Valid building blocks are listed below (some of these make more sense to use in a file name than others):

- <$FM_USER> - from *maker.ini* RegInfo/User
- <$FM_COMPANY> - from *maker.ini* RegInfo/Company
- <$FMX_USERNAME> - from *ditafmx.ini* Registration/Username
- <$FMX_FULLNAME> - from *ditafmx.ini* Registration/FullName
- <$OS_USERNAME> - %username% environment variable
- <$OS_COMPUTERNAME> - %computername% environment variable
- <$T_YYYY> - 4 digit year
- <$T_YY> - 2 digit year
- <$T_MM> - 2 digit month (zero padded)
- <$T_MON> - 3 character month
- <$T_MONTH> - full month name
- <$T_D> - 1 or 2 digit date
- <$T_DD> - 2 digit date (zero padded)
- <$T_HOUR> - 2 digit hour (zero padded)

- <$T_MIN> - 2 digit minute (zero padded)
- <$T_SEC> - 2 digit second (zero padded)
- <$TITLE> - the actual text of the title (as entered in the New File dialog)
- <$TITLE_LC> - the text of the title lowercased
- <$TITLE_NOSPACE> - the text of the title with spaces removed
- <$TITLE_NOSPACELC> - the text of the title, lowercased with spaces removed
- <$TITLE_NOSPACECAMEL> - the text of the title, camel-cased with spaces removed
- <$TITLE_NOSPACECAMELLOW> - the text of the title, camel-cased with spaces removed, and the first character lowercased
- <$TITLE_SPACETOUNDER> - the text of the title with spaces replaced with underscores
- <$TITLE_SPACETOUNDERLC> - the text of the title with spaces replaced with underscores and lowercased
- <$TOPIC_TYPE> - the topic type's element name
- <$UNIQUEID> - the unique ID as applied to the root topic element
- <$VAR(*VARNAME*)> - the value of the variable *VARNAME*

Note that you can include slashes (always use forward slashes or double backslashes in FrameMaker dialog boxes) in the New File Name Format field to automatically fill in subdirectory names.

You can include a modifier value following the building block name in square brackets. This value must be a number (from 0 to 99), and if provided, limits the length of the resulting string to that value (the first *N* characters). If you want to extract a substring from a building block, include the start and end positions in square brackets. For example, the following building block will extract the first two characters from the topic type:

```
<$TOPIC_TYPE[2]>
```

Or, to extract the second through fifth characters, use the following syntax:

```
<$TOPIC_TYPE[2-5]>
```

Other modifiers can be used to change the case of the text that results from the building block. These single-character modifiers must follow any numeric modifiers if present. The following modifiers are available:

- `U` - uppercase
- `L` - lowercase

- `T` - title case

The following syntax will generate the first two characters from the topic type in uppercase:

```
<$TOPIC_TYPE[2U]>
```

For example, if you always want files to be saved into folders based on the topic type, you might use the following format string:

```
<$TOPIC_TYPE>/<$TOPIC_TYPE[2]>_<$TITLE_NOSPACELC>.dita
```

If the title was “Using New Tools” and the topic type was task, the resulting filename would be “task/ta_usingnewtools.dita”.

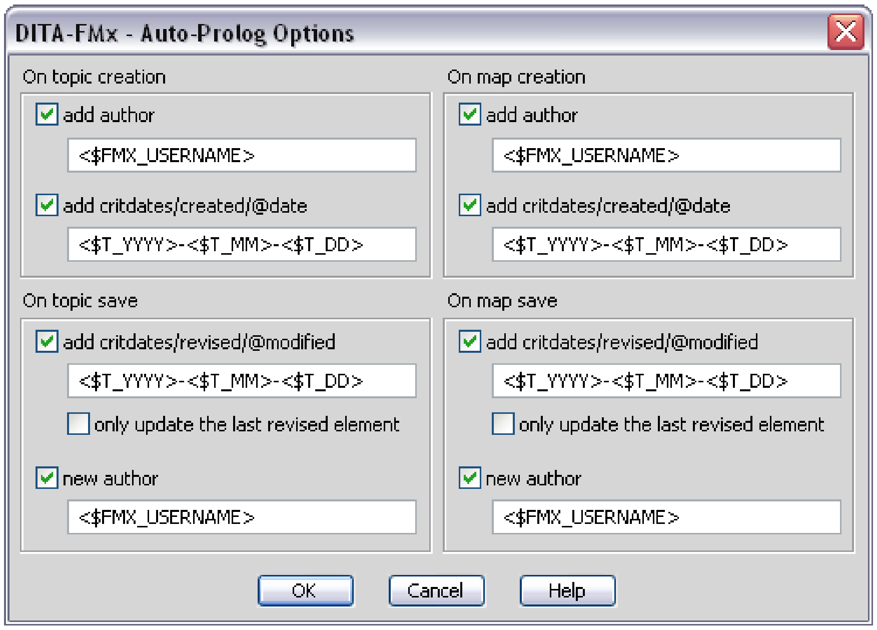
Related information:

"New DITA File" on page 1
"Creating Element Templates" on page 49

Auto-Prolog Options

Enable and specify the content for automatically inserted prolog data in topics and maps.

If enabled, the auto-prolog options automatically insert and update basic data in a topic’s <prolog> or a map’s <topicmeta> or <bookmeta>. There are two types of options, one for data that is added on file creation and another for data that is added/updated on file save. Each option can be enabled independently.


DITA-FMx - Auto-Prolog Options

On topic creation
- [✓] add author: <$FMX_USERNAME>
- [✓] add critdates/created/@date: <$T_YYYY>-<$T_MM>-<$T_DD>

On map creation
- [✓] add author: <$FMX_USERNAME>
- [✓] add critdates/created/@date: <$T_YYYY>-<$T_MM>-<$T_DD>

On topic save
- [✓] add critdates/revised/@modified: <$T_YYYY>-<$T_MM>-<$T_DD>
- [ ] only update the last revised element
- [✓] new author: <$FMX_USERNAME>

On map save
- [✓] add critdates/revised/@modified: <$T_YYYY>-<$T_MM>-<$T_DD>
- [ ] only update the last revised element
- [✓] new author: <$FMX_USERNAME>

Figure 3-18: DITA-FMx Auto-Prolog Options dialog box

Each option provides a text field where you can enter plain text and special building blocks (similar to those used for new file names, but limited in scope). The building blocks that are appropriate for this use are listed below.

File creation: add author

If enabled, inserts the value from this field into the <author> element. If an element template is used that contains an <author> element, this value is appended to that which already exists.

File creation: add critdates/created/@date

If enabled, inserts the value from this field into the critdates/created/@date attribute.

File save: add critdates/revised/@modified

If enabled, inserts the value from this field into the critdates/revised/@modified attribute.

If the **Only Update the Last Element** option is selected, the modified attribute value will be updated on the last critdates/revised element. If not selected, a new <revised> element is added each time the new value for the modified attribute is different than the previous sibling element (typically a new element for each day the file is saved).

File save: new author

If enabled, adds a new <author> element if different than the value of an existing <author> element. If multiple <author> elements exist and the current value matches an earlier value, that element is moved to be the last <author> element in the sequence.

The building blocks that are appropriate these fields are listed below (for details on the syntax, see Use of building blocks).

- <$FM_USER> - from *maker.ini* RegInfo/User
- <$FM_COMPANY> - from *maker.ini* RegInfo/Company
- <$FMX_USERNAME> - from *ditafmx.ini* Registration/Username
- <$FMX_FULLNAME> - from *ditafmx.ini* Registration/FullName
- <$OS_USERNAME> - %username% environment variable
- <$OS_COMPUTERNAME> - %computername% environment variable
- <$T_YYYY> - 4 digit year
- <$T_YY> - 2 digit year
- <$T_MM> - 2 digit month (zero padded)
- <$T_MON> - 3 character month

- <$T_MONTH> - full month name
- <$T_D> - 1 or 2 digit date
- <$T_DD> - 2 digit date (zero padded)
- <$T_HOUR> - 2 digit hour (zero padded)
- <$T_MIN> - 2 digit minute (zero padded)
- <$T_SEC> - 2 digit second (zero padded)

Related information:
"New DITA File" on page 1

Map Options

Navtitles in maps can be constructed in various ways.

Depending on your workflow, it may be important how and when <navtitle> elements are added to topicrefs in a map. These options control how navtitles are added.

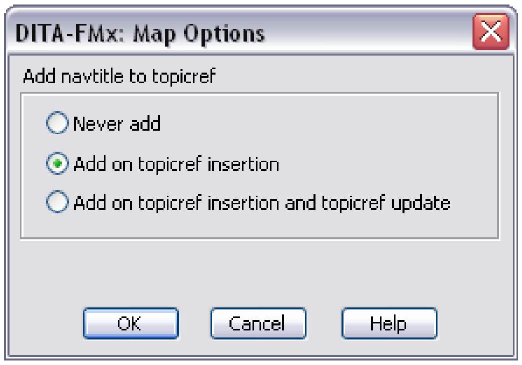

DITA-FMx: Map Options
Add navtitle to topicref
( ) Never add
(●) Add on topicref insertion
( ) Add on topicref insertion and topicref update

Figure 3-19: DITA-FMx Map Options dialog

With the "auto-load topicrefs" option enabled (in DITA Options), when authoring a map, a clickable label is provided for each topicref-based element. For topicrefs that have a <navtitle> element, the <navtitle> itself becomes that clickable label; if no <navtitle> is present, an <fm-reflabel> element is added as a temporary label. These are used as navigation aids for accessing topics referenced by a map structure.

Add Navtitle to Topicref

- **Never Add** - Just as it says; with this option enabled, navtitles are never added unless you specifically add them.
- **Add on Topicref Insertion** - When a topicref-based element is added to a map, the title of the associated file is added within the

<navtitle> and <topicmeta> elements. However, when a map is opened that has topicrefs without <navtitle> elements, the <fm-reflabel> element is used as a label.

- **Add on Topicref Insertion and Topicref Update** - In addition to navtitles being added on topicref insertion, when a map is opened, any topicrefs without a <navtitle> will be updated to include the topicmeta/navtitle structure.

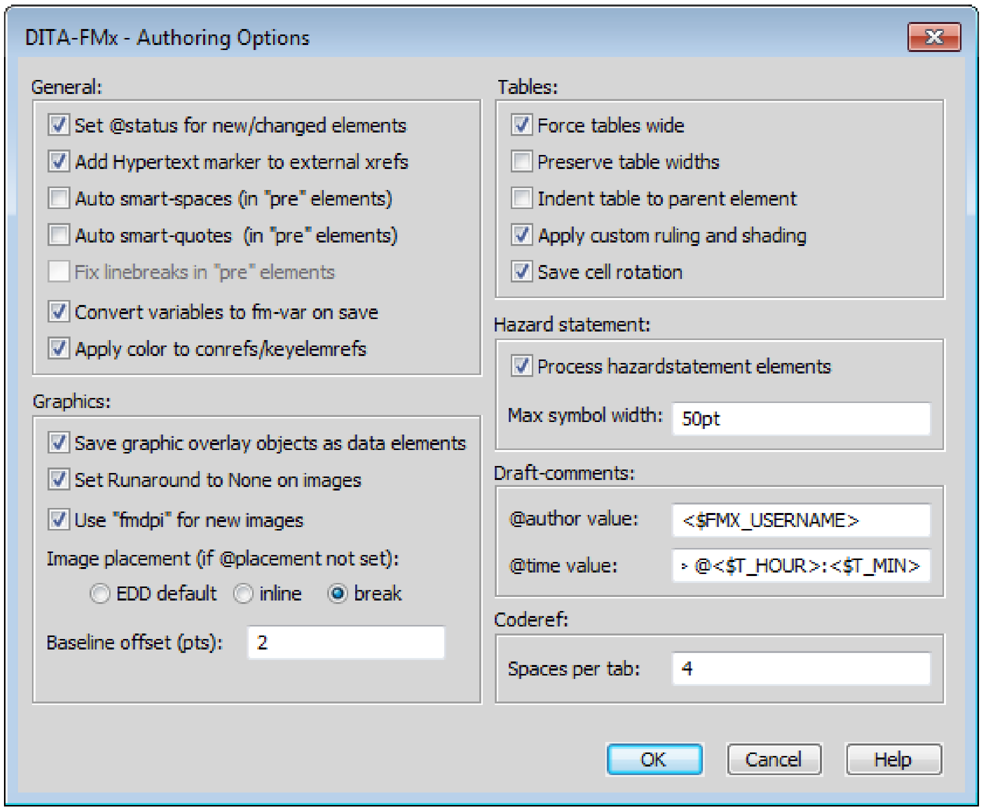
IMPORTANT: *Use of the "Add on Topicref Insertion and Topicref Update" option will cause topicmeta/navtitle structures to be added to all topicrefs in a map when it is opened and saved.*

RELATED INFORMATION:

"Using Keyspaces in DITA-FMx" on page 54
"Working with Keys" on page 56
"Keyspace Options" on page 51

Authoring Options

Provides control over the various settings available while editing DITA topics and maps.


DITA-FMx - Authoring Options

General:
- [✓] Set @status for new/changed elements
- [✓] Add Hypertext marker to external xrefs
- [ ] Auto smart-spaces (in "pre" elements)
- [ ] Auto smart-quotes (in "pre" elements)
- [ ] Fix linebreaks in "pre" elements
- [✓] Convert variables to fm-var on save
- [✓] Apply color to conrefs/keyelemrefs

Graphics:
- [✓] Save graphic overlay objects as data elements
- [✓] Set Runaround to None on images
- [✓] Use "fmdpi" for new images

Image placement (if @placement not set): EDD default inline [✓ break]

Baseline offset (pts): 2

Tables:
- [✓] Force tables wide
- [ ] Preserve table widths
- [ ] Indent table to parent element
- [✓] Apply custom ruling and shading
- [✓] Save cell rotation

Hazard statement:
- [✓] Process hazardstatement elements

Max symbol width: 50pt

Draft-comments:

@author value: <$FMX_USERNAME>

@time value: › @<$T_HOUR>:<$T_MIN>

Coderef:

Spaces per tab: 4

Figure 3-20: DITA-FMx Authoring Options dialog box

General

Set @status for New/Changed Elements

Automatically sets the value of the @status attribute on new and changed elements. For new elements, the @status attribute value is set to "new," and if an element's @status attribute has no value and you modify the content of that element, the @status attribute is set to "changed." Use the Reset @status command to delete the value from all @status attribute in the current file or workbook.

Add Hypertext Marker to External Xrefs

On file open or on insertion of an external xref (one that has the @scope attribute set to "external"), a FrameMaker Hypertext marker is added so that this element is hyperlinked when a PDF is generated through FrameMaker.

Use of this feature adds unstructured FrameMaker Hypertext markers to the document, which are saved to XML as processing instructions. This should not cause any problems for processing by the Open Toolkit or other tools.

Auto Smart-Spaces (in “pre” elements)

Toggles the FrameMaker Smart Space feature on and off as the insertion point is moved into and out of a preformatted element. A “preformatted” element is one that has a @class attribute value that contains “topic/pre” or “topic/lines”.

Auto Smart-Quotes (in “pre” elements)

Toggles the FrameMaker Smart Quotes feature on and off as the insertion point is moved into and out of a preformatted element. A “preformatted” element is one that has a @class attribute value that contains “topic/pre” or “topic/lines”.

Fix Line Breaks in “pre” Elements

In order to deal with a “bug” in FrameMaker, on file save this option adds a space between the end of an inline child element and the end of line in a preformatted element. A “preformatted” element is one that has a class attribute value that contains “topic/pre” or “topic/lines”.

If an inline child element starts at the beginning of a line within a preformatted element, this option moves the start element to the previous line and adds a space between it and the line end.

NOTE: This option is no longer needed as of FM12 and will not be available for selection in later versions.

Convert Variables to fm-var On Save

If enabled, FrameMaker variables are wrapped in an <fm-var> element which allows the variable to round-trip as a DITA ph element.

Apply Color to Conrefs and Keyelemrefs

Specifies that coloring is applied to conrefs and key element references in Topic files. If enabled, the color “DITA-Conref” is applied. This color should be defined in the template as a custom color. If this custom color is not defined and the option is selected, this color will be created and assigned the color Blue.

Graphics

Save Graphic Overlay Objects as data Elements

If enabled, graphic overlay objects (e.g., lines and callouts) placed over an image are stored as DITA <data> elements so they can be rebuilt when the file is reopened.

Set Runaround to None on Images

If enabled, the image Runaround property is set to “None” making it possible to place callouts over images.

Use “fmdpi” for New Images

Uses the “fmdpi” feature when inserting new raster images. If enabled, writes the value “fmdpi:*DPI*” to the @outputclass attribute of the <image> element (where *DPI* is replaced with the selected DPI value used to insert the image). This is a DITA-FMx feature that adjusts the size of the image to the DPI value specified when opened in FrameMaker; when processed by other means, the DPI value is ignored.

NOTE: Previous releases of DITA-FMx used the @otherprops attribute for this purpose which was considered to be non-standard. If you want to continue using the @otherprops attribute for this purpose, set the FmDpiUseOutputclass parameter to “1” in the GeneralExport section of the ditafmx.ini file.

Image Placement

Specifies the default value for the image @placement attribute if no value is set. Options are: EDD default, Inline, or Break.

Baseline Offset

Specifies the distance (in points) an inline image is shifted below the baseline of the surrounding text.

Tables

Force Tables Wide

If enabled, tables are forced to fill the text column (or page if the @pgwide attribute is set to 1). This overcomes an apparent bug in FrameMaker where under certain circumstances tables are not rendered full width in book builds. If your EDD already handles “pgwide” tables, you may need to disable this functionality but in general it should be enabled.

Preserve Table Widths

If enabled, the relative width (as a percentage) of each table is stored in the table’s @pgwide attribute (this only applies to <table> elements and specializations, <simpletable> specializations are not supported by this option). This value will be rounded to the nearest higher “5” increment. If this feature is enabled, DITA-FMx will interpret a @pgwide value greater than 1 to set the relative width of the table. If this option is used, we recommend also enabling the **Force Tables Wide** option.

Note that while the use of the @pgwide attribute for this purpose is not technically “invalid” (from the DITA perspective), it will not be honored by processors other than DITA-FMx so use it with caution. Also, use of this feature may require modification to the @pgwide attribute definition in your EDD to allow values other than 0 or 1.

Apply Custom Ruling and Shading

If enabled, applies custom ruling and shading to table rows or cells based on the @outputclass attribute. The custom ruling or shading is copied from corresponding row or cell elements in a table on the “fmx-table-format” reference page.

Save Cell Rotation

If enabled, table cell rotation will be preserved for DITA topics when rendered in FrameMaker with DITA-FMx.

Hazard statement

Process hazardstatement elements

Enables the processing of hazardstatement elements in the authoring view.

Max symbol width

Specifies the maximum width for hazardsymbol images in both the authoring and publishing views. This setting can be overridden for specific book deliverables with the BookBuildOverrides/HazardSymbol-MaxWidth setting in the book-build INI file. Note that this setting specifies a “maximum” width; it does not increase the width of smaller images.

Draft-Comments

@author Value

Defines the content that is added to the @author attribute when a <draft-comment> element is created. Enter plain text or use the building blocks listed for use with the Auto-Prolog feature.

@time Value

Defines the content that is added to the @time attribute when a <draft-comment> element is created. Enter plain text or use the building blocks listed for use with the Auto-Prolog feature.

Coderef

Spaces per Tab

Specifies the number of spaces that are added for each tab encountered in the content included by a <coderef>. This value must be greater than 0 and no more than 20.

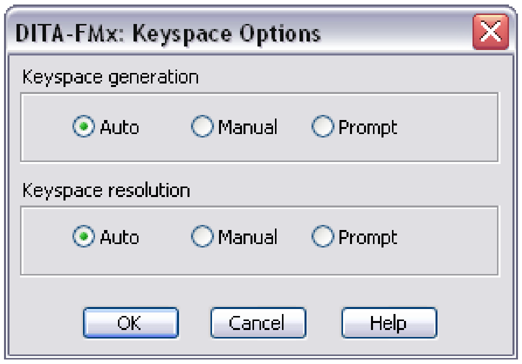
RELATED INFORMATION:

"Graphic Overlay Objects" on page 31
"Custom Ruling and Shading" on page 37
"Controlling Table Widths" on page 38
"Rotated Table Cells" on page 39

Keyspace Options

Options for controlling when a keyspace is rebuilt, and when the default keyspace changes.

When working with multiple projects and maps, it may be necessary to work with multiple keyspaces. These options allow you to work with keyspaces in a way that's most efficient for your needs.


DITA-FMx: Keyspace Options

Keyspace generation: [✓ Auto] Manual Prompt

Keyspace resolution: [✓ Auto] Manual Prompt

Figure 3-21: DITA-FMx Keyspace Options dialog

Keyspace Generation

Defines when the current map's keyspace is rebuilt, by scanning for key definitions in the root map and all submaps.

- **Auto** - Rebuilds the current map's keyspace each time that file is opened or saved. (Also rebuilds the keyspace when a submap has the <othermeta> "fmx-root-map" flag.)
- **Manual** - Never rebuilds the keyspace. Assumes that you use the Rebuild option in the Keyspace Manager dialog.
- **Prompt** - Prompts you to rebuild the current map's keyspace each time the map is opened or saved. (Also prompts to rebuild the keyspace when a submap has the <othermeta> "fmx-root-map" flag.)

Keyspace Resolution

Specifies when the "default" keyspace changes

- **Auto** - Detects if a newly opened map is registered with an associated keyspace. If it is, the default keyspace is automatically set to that map's keyspace.
- **Manual** - Never changes the default keyspace. Assumes that you change the default keyspace in the Keyspace Manager dialog.
- **Prompt** - Prompts you to approve changing the default keyspace when the newly opened map is registered with an associated keyspace.

If you are working with submaps and want these options to be triggered when editing a submap, add an <othermeta> element to the map's <topicmeta> where the @name attribute is set to "fmx-root-map" and the @content attribute is set to the relative path and file name of the root map.

For example, if the root map's name is "ditafmx.ditamap", add the following <othermeta> element to all submaps that define keys:

```
<othermeta name="fmx-root-map" content="ditafmx.ditamap"/>
```

Adding this flag to the submaps will cause the keyspace options to be triggered when editing the submaps.

NOTE: *If you're sharing submaps between multiple root maps, use of this feature may not produce the desired results.*

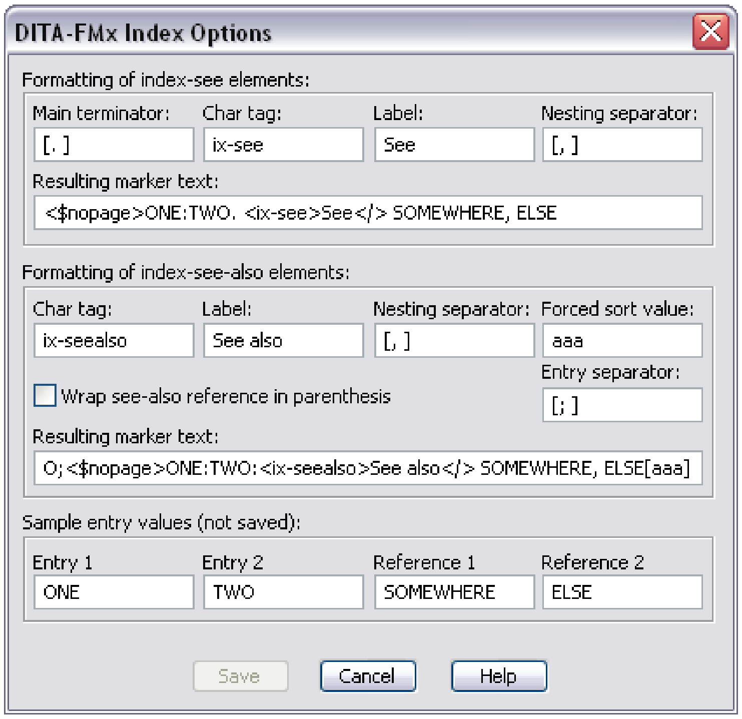
Related information:

"Using Keyspaces in DITA-FMx" on page 54
"Working with Keys" on page 56
"Map Options" on page 46

Index Options

Allows customization of the import/export processing of the marker syntax with index-see and index-see-also elements.

Although there are standards for the formatting of "see" and "see-also" index entries, each organization may want these entries to be constructed slightly differently. The **Index Options** dialog allows for this personalization. DITA provides the <index-see> and <index-see-also> elements for identifying these type of index entries, but in FrameMaker, the index syntax is all contained within a single Index marker (in DITA-FMx this becomes an <fm-indexterm> element). This dialog defines how the DITA elements map to and are converted from DITA into FrameMaker (and back).



Formatting of index-see elements:

Main terminator:	Char tag:	Label:	Nesting separator:
[.]	ix-see	See	[,]

Resulting marker text: <$nopage>ONE:TWO. <ix-see>See</> SOMEWHERE, ELSE

Formatting of index-see-also elements:

Char tag:	Label:	Nesting separator:	Forced sort value:
ix-seealso	See also	[,]	aaa

Entry separator: [;]

☐ Wrap see-also reference in parenthesis

Resulting marker text: O;<$nopage>ONE:TWO:<ix-seealso>See also</> SOMEWHERE, ELSE[aaa]

Sample entry values (not saved):

Entry 1	Entry 2	Reference 1	Reference 2
ONE	TWO	SOMEWHERE	ELSE

Figure 3-22: DITA-FMx Index Options dialog box

The Index Options dialog provides two main sections, one for each of the “see” and “see-also” entry types. Within each section are fields (described below) and a “Resulting Marker Text” area that dynamically shows the index marker syntax to be generated using the fields as entered. The textual content used for the sample entries can be modified in the Sample Entry Values field. Any time a value requires leading or trailing spaces, wrap it in square or curly brackets (“[..]” or “{ .. }”).

index-see Options

Main Terminator

Separates the text of the main entry with that of the referenced entry. Typically a period followed by a space.

Char Tag

The name of the character style used to format the “see” reference entry. This is required even if there is no additional formatting to be applied because it is the only way for FrameMaker to know if the index entry is a “see” entry. The character tag “ix-see” is defined in the default DITA-FMx templates; you are free to modify the formatting of that tag. If you want to use a different character tag, you must add that style to the template.

Label

The label that separates the main entry and the referenced entry text. This value will be formatted with the character style specified in the Char Tag field.

Nesting Separator

Separates multiple referenced entries. Typically a comma with a trailing space.

index-see-also Options

Char Tag

The name of the character style used to format the “see-also” reference entry. This is required even if there is no additional formatting to be applied because it is the only way for FrameMaker to know if the index entry is a “see-also” entry. The character tag “ix-seealso” is defined in the default DITA-FMx templates; you are free to modify the formatting of that tag. If you want to use a different character tag, you must add that style to the template.

Label

The label that separates the main entry and the referenced entry text. This value will be formatted with the character style specified in the Char Tag field.

Nesting Separator

Separates multiple referenced entries. Typically a comma with a trailing space.

Forced Sort Value

If you want the entry to sort in a particular manner, enter the forced sort text here.

Entry Separator

String that separates multiple top-level target entries. Typically a semi-colon with a trailing space. This value is not represented in the sample Resulting Marker Text field.

Wrap See-Also Reference

If enabled, wraps the entire see-also reference in parenthesis.

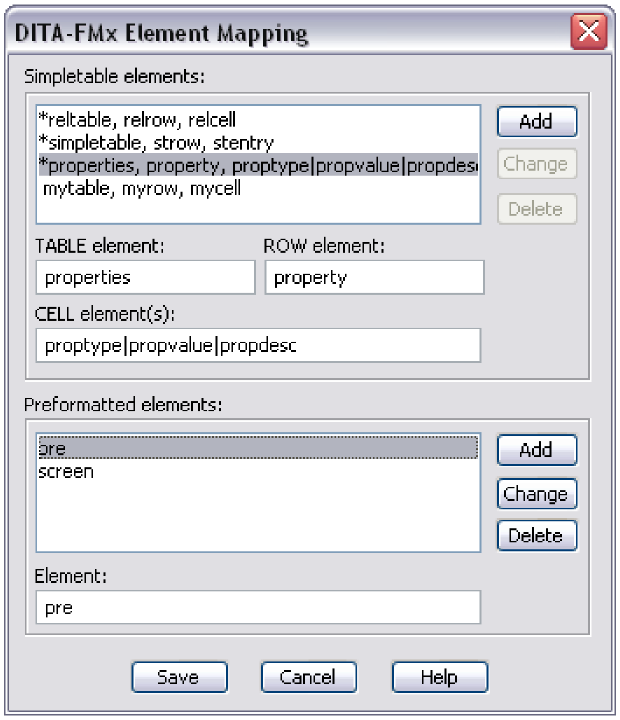
RELATED INFORMATION:

"Working with Indexterms" on page 47

Element Mapping

Allows the specification of simpletable element types and preformatted element types. Any specializations of these element types should be added here.

The Element Mapping dialog lists the default and specialized elements that are processed in a particular way by DITA-FMx.



*reltable, relrow, relcell
*simpletable, strow, stentry
*properties, property, proptype|propvalue|propdes
mytable, myrow, mycell

TABLE element: properties
ROW element: property
CELL element(s): proptype|propvalue|propdesc

pre
screen

Element: pre

Figure 3-23: DITA-FMx Element Mapping dialog box

When <simpletable>-based elements are encountered during the import process, FrameMaker needs to be able to count the number of columns in each table. This information is typically stored in attributes within the table element, but the DITA specification does not provide this type of attribute for <simpletable>s (and any table based on a <simpletable>).

Using the information provided in this dialog, FrameMaker can determine the number of columns in these tables. The default <simpletable> elements are indicated with an asterisk. Even though the <choicetable> elements are specialized from <simpletable>, they are not listed in this dialog because they must always have two columns and are therefore defined in the read/write rules file.

If you've specialized <simpletable> to create one of your own, you'll need to add it to this list. Enter the table, row, and cell element names in the fields provided. If your table uses different element names for the cells (like the <properties> table), you can enter the multiple names separated by the vertical bar character.

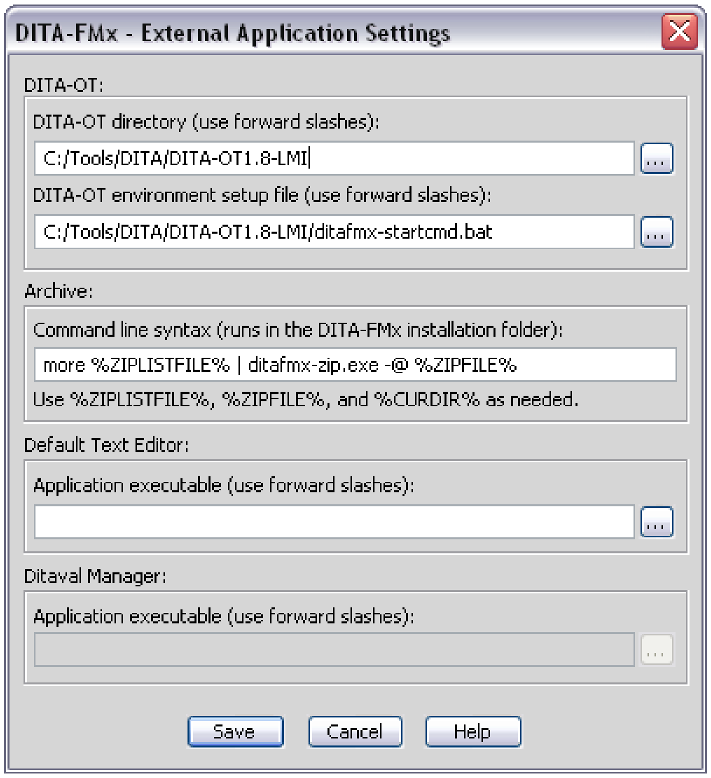
Related information:

"Working with Tables" on page 35

External Application Settings

Specifies the path and file name of applications called from DITA-FMx as well as any options passed to those applications.

The **External Application Settings** dialog allows you to specify the location of the associated application and any command line syntax or parameters. All paths entered into this dialog must use the slash as the directory delimiter.


DITA-FMx - External Application Settings

DITA-OT:

DITA-OT directory (use forward slashes): C:/Tools/DITA/DITA-OT1.8-LMI

DITA-OT environment setup file (use forward slashes): C:/Tools/DITA/DITA-OT1.8-LMI/ditafmx-startcmd.bat

Archive:

Command line syntax (runs in the DITA-FMx installation folder): more %ZIPLISTFILE% | ditafmx-zip.exe -@ %ZIPFILE%

Use %ZIPLISTFILE%, %ZIPFILE%, and %CURDIR% as needed.

Default Text Editor:

Application executable (use forward slashes):

Ditaval Manager:

Application executable (use forward slashes):

Figure 3-24: DITA-FMx External Application Settings dialog box

DITA-OT Directory

Specifies the main DITA-OT (DITA Open Toolkit) directory. (Required in order to use the **Generate Output** command.)

DITA-OT Environment Setup File

Specifies the DITA-OT environment setup batch file. (Required in order to use the **Generate Output** command.)

For OT 2.x or later, this field specifies the *bin\dita.bat* file in the DITA-OT installation.

For OT 1.x, this file is a copy (with minor modifications) of the default *startcmd.bat* file provided with the OT.

Archive

Specifies the command line syntax that is used to create the archive. Before creating the archive, the list of files is generated based on the current file and all referenced files (as well as the archive baggage file described in the Create Archive topic). This file is written to the user’s DITA-FMx folder (**DITA-FMx > Open DITA-FMx Folder**) as *~tmpzipfiles.txt*. The generated archive is named *<filename>_zip.zip*, where *<filename>* is the root file name of the current file when the Create Archive command is used. For example, if the archive is created from the *project_a.ditamap* file, the archive created will be named *project_a_zip.zip*.

There are three variables that can be used in this field, %ZIPLISTFILE%, %ZIPFILE%, and %CURDIR%. The string “%ZIPLISTFILE%” is replaced with the path and filename of the list of files to archive. The string “%ZIPFILE%” is replaced with the path and file name of the archive file name. The string “%CURDIR%” is replaced with the path to the current file.

This command line is included in a batch file created in the user’s DITA-FMx folder as the file *~tmpzip.bat*. before the command line is executed, the current directory is set to DITA-FMx installation folder (typically in the Program Files area). Any commands you place in this field should be designed to run from that folder.

The default command line syntax is the following.

```
more %ZIPLISTFILE% | ditafmx-zip.exe -@ %ZIPFILE%
```

This command syntax passes the content of the archive file list (%ZIPLISTFILE%) to the ditafmx-zip utility (the -@ option tells the utility to read from “stdin”). It then generates the archive file specified by the %ZIPFILE% variable.

*If the archive is not created, verify that the Command Line Syntax value in the DITA Options: External Applications dialog is valid. Deleting this value will reset it to the default value. You can test by running the ~tmpzip.bat batch file in the user’s DITA-FMx folder (**DITA-FMx > Open DITA-FMx Folder**); run this from a command shell to see any errors that may display.*

Default Text Editor

Specifies the application used to edit a <coderef> when the Text Editor button is selected in the Coderef Manager dialog. Enter the full path and file name to the application executable file.

Ditaval Manager

This feature is not currently enabled.

RELATED INFORMATION:

"Generate Output" on page 78
"Create Archive" on page 7

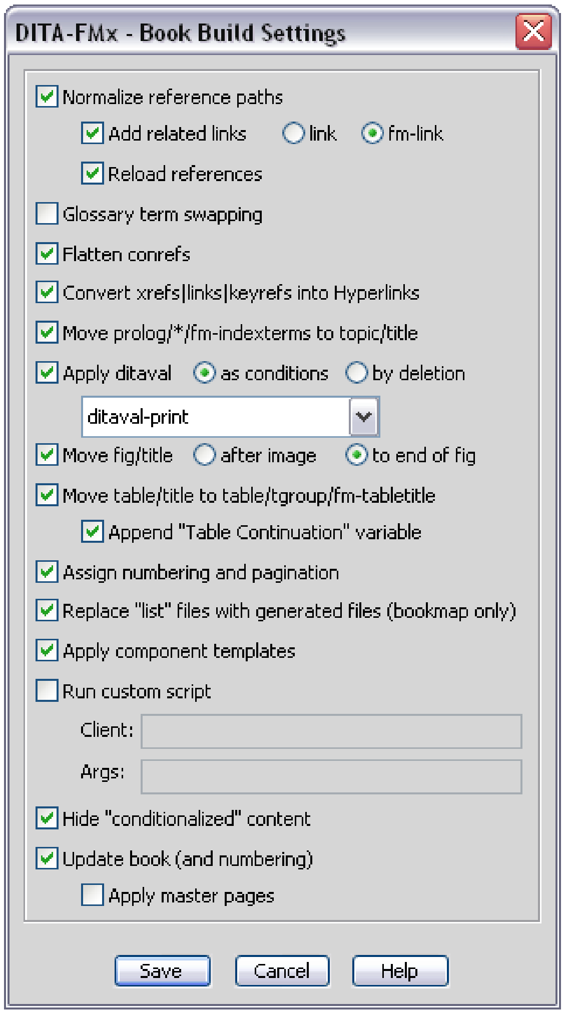
Book Build Settings

Specifies the automated processes to be run on a FrameMaker book that has been generated from a DITA map.

The **Generate Book from Map** command aggregates all of the topic files referenced by a DITA map (and submaps) into a FrameMaker book and chapter files. After the book has been assembled, a number of processes can be run on the book and book components to prepare it for publishing. The Book Build Settings dialog allows you to specify which of these processes are performed.

The settings in this dialog can be overridden by settings in the BookBuildOverrides section of a *ditafmx-bookbuild.ini* file (also known as a “book-build INI file”). This dialog should be used for testing of various options, but to ensure consistent book generation, you should set up a book-build INI file for each project.

NOTE: *The term “chapter file” is used in a generic sense and does not imply chapter versus appendix or other book component type. It just means a “FM binary file”.*



- [✓] Normalize reference paths
  - [✓] Add related links ( ) link (●) fm-link
  - [✓] Reload references
- [ ] Glossary term swapping
- [✓] Flatten conrefs
- [✓] Convert xrefs|links|keyrefs into Hyperlinks
- [✓] Move prolog/*/fm-indexterms to topic/title
- [✓] Apply ditaval (●) as conditions ( ) by deletion
  - ditaval-print
- [✓] Move fig/title ( ) after image (●) to end of fig
- [✓] Move table/title to table/tgroup/fm-tabletitle
  - [✓] Append "Table Continuation" variable
- [✓] Assign numbering and pagination
- [✓] Replace "list" files with generated files (bookmap only)
- [✓] Apply component templates
- [ ] Run custom script
  - Client:
  - Args:
- [✓] Hide "conditionalized" content
- [✓] Update book (and numbering)
  - [ ] Apply master pages

Figure 3-25: DITA-FMx Book Build Settings dialog box

The options in this dialog specify the processes that are run on the book and are shown in the order that they are run.

Normalize Reference Paths

Iterates over all chapter files and converts all @href and @conref attribute values into absolute paths.

Add Related Links

Adds appropriate links to <related-links> elements in each topic based on the <reltable>(s) in the DITA map. You can select the link type of “link” or “fm-link” (standard DITA links or FM-based cross-refs). If you use fm-links, the “RelatedLink” cross-reference format is used. This option is only available if the Normalize Reference Paths option has been selected.

Reload References

Iterates over all references in all chapters and updates/reloads them based on the rebuilt reference paths. This option is only available if the Normalize Reference Paths option has been selected.

Glossary Term Swapping

If enabled, instead of using the <glossterm> element as the content for a keyref to a glossary entry, the <glossSurfaceForm> element is used for the first instance of a term in a chapter, and the <glossAlt> elements are used for later instances.

Flatten Conrefs

Runs the **Flatten Conrefs** command. This is useful if you have conrefs that contain cross-refs (<xref>s, <link>s, <fm-xref>s, or <fm-link>s) since this will allow those cross-refs to become live hyperlinks.

Convert Xrefs/Links into Hyperlinks

Runs the **Xref to Hyperlink** command. This is only useful if you your files contains DITA <xref> or <link> elements. This only processes internal xrefs/links; if you need to make external xrefs into live links, you'll need to enable the Add Hypertext Marker to External Xrefs option before converting the map to a book.

Move prolog/*/fm-indexterms to topic/title

Moves all of the <fm-indexterm> elements in the prolog to the end of the parent topic's <title> element. If this option is not selected, any <index-term> elements in the <prolog> will not be included in a generated index if the <prolog> is hidden.

Apply Ditaval

Runs the **Apply Ditaval** command on the generated book file. Select the **As Conditions** or **By Deletion** option. Select a registered ditaval file from the list. Use the **Ditaval Manager** to register a ditaval file with DITA-FMx.

*IMPORTANT: When using the **As Conditions** option, this feature uses the default settings as defined in the Apply Ditaval dialog box. In order to change the way the conditions are applied, you must run the **Utilities > Apply Ditaval** command manually before using this option.*

Move fig/title to End of fig Element

Moves each <title> of a <fig> element to the end of the <fig> element. If a <desc> element follows the <title>, it (and any child elements) will be moved as well. Essentially moves the figure title so it follows the image. For this option to work properly, the Book application must have the title element valid at the end of a <fig> element's general rule.

Move table/title to table/tgroup/title

Moves each <title> of a <table> element to the location in the FrameMaker object structure that allows it to work as a FM table title is supposed to (appear at the top of the table on new pages). In the FM object model, the <tgroup> element is actually the "table", while the <table> element is just a container that wraps the table.

This option moves the DITA table title so that it works properly in the generated FrameMaker files. If you select the **Append 'Table Continuation' variable** option, the Table Continuation variable will be appended to the table title after it is moved to the new location. To modify the text or format of the Table Continuation variable, edit the topic template file. For this option to work properly, the Book application must have the <fm-tabletitle> element defined and valid as a child of the <tgroup> element.

Assign Numbering and Pagination

Sets up the book component's numbering properties based on the settings in the *ditafmx-bookbuild.ini* file. DITA-FMx looks for this INI file in the directory that the book is built in, then in the user's DITA-FMx folder (**DITA-FMx > Open DITA-FMx Folder**).

Sections named "NumberingFirst-<*maptype*>" and "NumberingDefault-<*maptype*>" define the numbering and pagination properties for the first and remainder files created from specific map elements (<*maptype*> refers to the element name of the top-level map element).

Replace List Files with Generated Files

Replaces any "list" files with the appropriate corresponding FrameMaker generated list. For example, the frontmatter/booklists/toc entry will be replaced with a generated TOC, and the backmatter/booklists/indexlist entry will be replaced with a generated index.

This replacement relies on settings in the *ditafmx-bookbuild.ini* file. DITA-FMx looks for this INI file in the directory that the book is built in, then in the user's DITA-FMx folder (**DITA-FMx > Open DITA-FMx Folder**). In the General section of the *ditafmx-bookbuild.ini* file the BookTemplatesDir parameter indicates the directory that contains the template files used as the basis for generated book components. These files should be named *gentpl~<maptype>.fm*, where <maptype> is the name of the associated map element that should be replaced with a generated file. Other sections named "GeneratedFile-<maptype>" define the component type and other properties needed for this file replacement.

Apply Templates

Applies templates (for both formatting and element definitions) to each of the book components based on the settings in the *ditafmx-bookbuild.ini* file. DITA-FMx looks for this INI file in the directory that the book is built in, then in the user's DITA-FMx folder (**DITA-FMx > Open DITA-FMx Folder**).

In the General section of the *ditafmx-bookbuild.ini* file the BookTemplatesDir parameter indicates the directory that contains the template files used for this process. These files should be named *tpl~<maptype>.fm*, where <maptype> is the name of the associated map element that should be updated with the specified template.

Run Custom Script

Runs a custom FDK client, FrameScript, or ExtendScript (FM10 and later only). Enter the client name, and any arguments that are to be passed to the client. If you'd like to pass the FDK ID of the book to the client, enter "%BOOKID%" in the Args field.

To run a FrameScript, enter "fsl" as the client name, and to run an ExtendScript (with FM10 and later) enter "ScriptingSupport" as the client name. For a FrameScript, the script name is the argument and for an ExtendScript provide the script file name as the argument.

You can run multiple clients by separating the client name and arguments with the vertical bar character. If you are specifying multiple clients, the first character in both the client list and the arguments list must be a vertical bar. Be sure that the number of vertical bars in the client list and argument list are the same to ensure that the arguments are passed to the proper client. If no arguments are needed for certain clients but are needed for others, you can enter a hyphen where no arguments are needed. For example, the following client and argument data will run three FDK clients:

Client: `| client1 | client2 | client3`

Args: `| client1-arg %BOOKID% | client2-arg | client3-arg %BOOKID%`

NOTE: *This option runs scripts before "pagination" has been completed. Conditional text is still shown and other formatting processes are yet to be run that will affect pagination. If you need to run a script after pagination has been completed, use the RunPostPaginationScript book-build INI setting, described in Book-Build INI file.*

Hide "Conditionalized" Content

Hides the elements that have been tagged through any of the four "Conditionalize" options in the Options dialog.

Update Book

Runs the **Edit > Update Book** command with all of the "update" options enabled except "Apply Master Pages". If you'd like to apply master pages, select the **Apply Master Pages** option. The Apply Master Pages option assumes that your template (or component templates) provide a Struct-MasterPageMaps table.

IMPORTANT: *On FM versions prior to FM10, the* ***Apply Master Pages*** *option will result in a message box, warning you that all current master pages will be reapplied. This interrupts any automation, and pauses the build until you choose the "Yes" button. Also, on all FM versions if this option is enabled and no StructMasterPageMaps table exists in the template(s) in use, an alert message displays.*

Related information:

"Book-Build INI file" on page 62
"Setting up Book Builds (PDF)" on page 63

Book-Build INI file

Controls all aspects of generating FM files from a DITA map using the Generate Book from Map command.

The *ditafmx-bookbuild.ini* file defines many of the parameters used to assemble the generated book to match your needs. The book build process first looks for this file in the directory that you have specified for the new book file, then it checks in the user's DITA-FMx folder (**DITA-FMx > Open DITA-FMx Folder**).

If you always use the same settings, you can just keep one copy in the user's DITA-FMx folder, but if you have book-specific settings, you may want to add this file to each output directory. This file follows the standard INI file format with section names in square brackets and key/value pairs following each section using a *key=value* syntax.

The book build process operates on individual book components based on the names of the map elements that are referencing the topic files (such as "part," "chapter," "toc," "indexlist," and so on). These element names are stored in the @mapelemtype attributes on the <fm-ditafile> elements in the generated book. The *ditafmx-bookbuild.ini* file provides two general section types, one that defines the numbering and pagination properties for each element type and one that defines the creation of generated FM files that replace the "list" files in the <frontmatter> and <backmatter> elements. The basic syntax for this file is as follows:

```
[General]
BookTemplatesDir=<path to templates folder>
PreBuildScript=<c:\projects\prebuild.bat %MAPNAME%>
PostBuildScript=<c:\projects\postbuild.bat %MAPNAME%>
UseOutputclassForType=(0|1)
ImportAttrsAsVars=(0|1)
ImportAttrsAsConds=(0|1)
```

```
AttrAsVarPrefix=<your_prefix>
AttrAsCondPrefix=<your_prefix>
AttrAsCondDefaultState=(Show|<Hide>)
ShowBuildTimes=(0|1)

[PdfDocInfo]
; maps field to fm-ditabook attribute value (starts with "@") or string
Author=@authorname
Title=@title
Subject=this is the subject
Keywords=keyword one, keyword two
Copyright=
Web Statement=
Job=
Marked=

[AltBookTemplatesDirs]
Count=<N>
1=<some path>
2=<another path>
<N>=<as needed>

[ConditionMap]
;fmx-<attribute>-<value>=(Show|Hide)

[DitavalDefaultsOverrides]
;DitavalName=
;DitavalExcludeConditionName=
;DitavalExcludeConditionNameType=
;DitavalExcludeConditionVisibility=
;DitavalFlagConditionName=
;DitavalFlagConditionNameType=
;DitavalFlagConditionVisibility=

[NumberingFirst-<mapelemtype>]
... (same as below)

[NumberingDefault-<mapelemtype>]
VolumeProperty=(Restart|Continue|UseSame|FromFile)
VolumeNumberValue=<value>
VolumeNumberFormat=(<see valid values below>|Text)
ChapterProperty=(Restart|Continue|UseSame|FromFile)
ChapterNumberValue=<value>
ChapterNumberFormat=(<see valid values below>|Text)
PageProperty=(Restart|Continue|FromFile)
PageNumberValue=<value>
PageNumberFormat=(<see valid values below>|Text)
ParagraphProperty=(Restart|Continue|FromFile)
FootnoteProperty=(Restart|StartOver|Continue|FromFile)
FootnoteNumberValue=<value>
FootnoteNumberFormat=(<see valid values below>|Custom)
FootnoteNumberCustom=<value>
TableFootnoteProperty=(Format|FromFile)
TableFootnoteNumberFormat=(<see valid values below>|Custom)
```

```
TableFootnoteNumberCustom=<value>
PageStartSide=(FromFile|Next|Left|Right)
PageDoubleSided=(1|0)
PageRounding=(DeleteEmpty|MakeEven|MakeOdd|NoChange)

[GeneratedFile-<mapelemtype>]
ComponentType=(IndexAuthor|IndexFormats|IndexMarker|IndexReferences|IndexStandard|In
dexSubject|ListFigure|ListFormat|ListMarker|ListMarkerAlpha|ListPara|ListParaAlpha|L
istReferences|ListTable|Toc)
NumTags=<N>
1=<paratag or markername>
2=<paratag or markername>
<N>=<paratag or markername>

[IncludeFiles]
<position>=<filename>

[IncludeFileTypes]
<position>=<mapelemtype>

[BookBuildOverrides]
NormalizeRefs=(0|1)
AddRelLinks=(0|1)
RelLinkType=(0|1)
ReloadRefs=(0|1)
GlossTermSwapping=(0|1)
XrefToHyperlink=(0|1)
ApplyDitaval=(0|1)
DitavalName=<registered ditaval name or file name>
DitavalFilterType=(0|1|2)
MovePrologIndex=(0|1)
MoveFigTitles=(0|1)
MoveTableTitles=(0|1)
AppendTableContVar=(0|1)
FlattenConrefs=(0|1)
AssignNumbers=(0|1)
ReplaceListFiles=(0|1)
ApplyTemplates=(0|1)
RunScript=(0|1)
ScriptName=<script names>
ScriptArgs=<script arguments>
HideConditionalizedContent=(0|1)
UpdateBook=(0|1|2)

; The following parameters are not exposed in the UI, must be set in INI
RunPostPaginationScript=(0|1)
PostPaginationScriptName=<script names>
PostPaginationScriptArgs=<script arguments>
MoveFigId=(0|1)
MoveTableId=(0|1)
BreakToInline=(0|1)
NormalizeConditions=(0|1)
AssignMasterPagesFromMarkers=(0|1)
RetagElements=<element list>
```

```
FilterByAttribute=BUILD-EXPRESSION>CONDITION-NAME
BuildImagemapHotspots=(0|1)
FixTableWidths=(0|1)
ShowStatus=(0|1)
HazardToTable=(0|1)
HazardSymbolMaxWidth=<value>
UseColorFromTable=(0|1)
```

Valid values for the *NumberFormat entries are: Numeric, LCRoman, UCRoman, LCAlpha, UCAlpha, Kanji, Zenkaku, UCZenkaku, LCZenkaku, Kazu, Daiji, FullWidthNumeric, UCFullWidthAlpha, LCFullWidthAlpha, ChineseNumeric, and Text or Custom.

Each section and item are described below.

General Section

BookTemplatesDir

Specifies the directory that contains the component templates. There are two types of component templates: layout templates (*tpl~*.fm*) and generated file templates (*gentpl~*.fm*) files. You can include the $STRUCTDIR "variable" to specify the FrameMaker Structure directory. A relative path is relative to the generated book file.

ImportAttrsAsVars

If set to 1, results in FrameMaker variables being created or updated in all book components (both structured and unstructured) where the variable name is "<*prefix*><*attributename*>" and the value of the variable is set to the attribute value. The default <*prefix*> value is "fmx-", but can be changed or set to an emptying string with the AttrAsVarPrefix parameter, described below.

These variables can be used as header/footer variables or elsewhere in a document's layout. If the variable exists in the template or predefined FM file, its value will be updated, otherwise the variable will be created. For more information see, Adding Map to Book Metadata Mappings.

ImportAttrsAsConds

If set to 1, results in the hide/show state of FrameMaker conditions being set in all book components (both structured and unstructured) where the condition name is "<*prefix*><*attributename*>-<*attributevalue*>". The default <*prefix*> value is "fmx-", but can be changed or set to an emptying string with the AttrAsCondPrefix parameter, described below.

Only conditions that exactly match the composite names are set. The hide/show state of the conditions is determined by the names in the ConditionMap section (described below). This feature works best for attributes that use enumerated values as the attribute value. For example, you may enable an <fm-ditabook> attribute named "permission", and one of the possible values is “controlled”. You would create a condition named “fmx-permission-controlled” and tag the text “CONTROLLED” in the footer with this condition. The default state of this condition may be hidden, and if the attribute is set to “controlled”, the condition is toggled to show. For more information see, Adding Map to Book Metadata Mappings.

AttrAsVarPrefix

If provided, this parameter specifies the prefix of variables created when imported from map metadata (if the ImportAttrsAsVars feature is enabled). If not provided, the default value of “fmx-” is used as the prefix. To create variables with no prefix, provide this parameter with no value.

AttrAsCondPrefix

If provided, this parameter specifies the prefix of conditions defined in the ConditionMap section (if the ImportAttrsAsConds feature is enabled). If not provided, the default value of “fmx-” is used as the prefix. To create conditions with no prefix, provide this parameter with no value.

UseOutputclassForType

If set to 1, uses the value of the top-level topicref-based element’s @outputclass attribute to specify the component template’s “mapelem-type” value. If no @outputclass value is set, the element type will be used. This allows you to specify different component templates for files of the same map element type. If this option is enabled, you’ll need to include the corresponding NumberingFirst/NumberingDefault sections for that value. For example, if you set the @outputclass of the first <appendix> element to “appxspecial”, you’ll need to include a section labeled “NumberingFirst-appxspecial”.

NOTE: *If you are using submaps, you’ll need to set this @outputclass value on the root “topicref” element in the submap rather than the referencing element in the root map. This limitation may be addressed in a future update to DITA-FMx.*

PreBuildScript

If provided, this parameter specifies the name of a batch file or executable script that will run before any book-build processing is begun (this is not used to run an FDK client or FrameScript). You can use this to copy or rename files that may vary under different conditions or perform other pre-build setup tasks. Two “variables” are available to provide information to the script. The string “%MAPNAME%” is replaced with the DITA map name (no path or file extension) and “%MAPFILE%” is replaced with the full path and file name of the DITA map.

PostBuildScript

If provided, this parameter specifies the name of a batch file or executable script that will run after all book-build processing has completed (this is not used to run an FDK client or FrameScript). You can use this to copy or rename files that may vary under different conditions or perform other post-build tasks. Two “variables” are available to provide information to the script. The string “%MAPNAME%” is replaced with the DITA map name (no path or file extension) and “%MAPFILE%” is replaced with the full path and file name of the DITA map.

ShowBuildTimes

This option prints the current build time of each step in the book-build process, which can help in debugging. To enable this option, add a ShowBuildTimes=1 entry to the General section. A value of “1” enables the option and a value of “0” disables it (the default).

PdfDocInfo Section

The PdfDocInfo section defines the values assigned to fields in the PDF Document Info dictionary of the generated book file. Each key and value pair in this section are added to the book and then passed to the PDF when it is created. The standard fields are shown below, but you can use any fields that you feel are needed.

```
[PdfDocInfo]
Author=@authorname
Title=@title
Subject=this is the subject
Keywords=keyword one, keyword two
Copyright=
Web Statement=
Job=
Marked=
```

If the value starts with an “@” symbol, the corresponding attribute value is extracted from the fm-ditabook element in the generated book. Otherwise the specified content is written to the PDF info dictionary.

AltBookTemplatesDirs Section

The AltBookTemplatesDirs section is used to define alternate component template folders available as overrides in the Generate Book from Map dialog. To use this feature, set the Count parameter to the number of alternate paths to display, then add sequential numeric entries for each path.

```
[AltBookTemplatesDirs]
Count=2
1=C:/some/path
2=C:/another/path
```

You can add as many paths to this section as you'd like. They display as a list in the Generate Book from Map dialog. The "default" setting is defined by the path specified in the BookTemplatesDir parameter.

ConditionMap Section

The ConditionMap section is used to assign a hide/show state to conditions that are set based on the value of <fm-ditabook> attributes. If the General/ImportAttrsAsConds parameter is set to 1, the values in the ConditionMap section are read.

Each entry in the ConditionMap section specifies the full composite condition name and the hide/show state to set that condition. To use the example described above (in the ImportAttrsAsConds description), the following ConditionMap section would be used to show the "CONTROLLED" label in a footer if the permissions attribute is set to "controlled".

```
[ConditionMap]
fmx-permission-controlled=Show
```

You can add as many conditions to this section as you'd like, each is used only if the attribute name and value match a defined entry. These conditions are updated in the FM book components only if the condition exists in that file.

Use the General/AttrAsCondPrefix parameter to specify a prefix other than "fmx-".

DitavalDefaultsOverrides Section

The DitavalDefaultsOverrides section can be used to override the values in the DitavalDefaults section of the *ditafmx.ini* file. These are the values that are set when you configure the options in the **Apply Ditaval** dialog. This only applies if you are using the "filter by conditions" option.

```
[DitavalDefaultsOverrides]
;DitavalName=
;DitavalExcludeConditionName=
;DitavalExcludeConditionNameType=
;DitavalExcludeConditionVisibility=
;DitavalFlagConditionName=
;DitavalFlagConditionNameType=
;DitavalFlagConditionVisibility=
```

This section is used if the DitavalName key has a value assigned. The easiest way to set this up is to open the ditafmx.ini file then copy and paste the content from the DitavalDefaults section into the DitavalDefaultsOverrides section of the book-build INI file. Manually edit the values as needed. Doing this ensures that the proper conditional settings are used regardless of the default settings that may be defined.

Numbering(First|Default)-<*mapelemtype*> Sections

A NumberingFirst-<*mapelemtype*> section applies to the first occurrence of this element in the map, and NumberingDefault-<*mapelemtype*> applies to any additional like-named elements in the map.

VolumeProperty

Correlates to the options on the **Volume** tab of the Numbering Properties dialog. Valid values are: Restart, Continue, UseSame, and FromFile.

VolumeNumberValue

If Restart is specified for VolumeProperty, defines the number to use as the starting value (use numeric values only regardless of the number format).

VolumeNumberFormat

If Restart is specified for VolumeProperty, defines the format for that number and all subsequent numbers of this type that follow. Valid values are: Numeric, LCRoman, UCRoman, LCAlpha, UCAlpha, Kanji, Zenkaku, UCZenkaku, LCZenkaku, Kazu, Daiji, FullWidthNumeric, UCFullWidthAlpha, LCFullWidthAlpha, ChineseNumeric, and Text.

ChapterProperty

Correlates to the options on the **Chapter** tab of the Numbering Properties dialog. Valid values are: Restart, Continue, UseSame, and FromFile.

ChapterNumberValue

If Restart is specified for ChapterProperty, defines the number to use as the starting value (use numeric values only regardless of the number format).

ChapterNumberFormat

If Restart is specified for ChapterProperty, defines the format for that number and all subsequent numbers of this type that follow. Valid values are: Numeric, LCRoman, UCRoman, LCAlpha, UCAlpha, Kanji, Zenkaku, UCZenkaku, LCZenkaku, Kazu, Daiji, FullWidthNumeric, UCFullWidthAlpha, LCFullWidthAlpha, ChineseNumeric, and Text.

PageProperty

Correlates to the options on the **Page** tab of the Numbering Properties dialog. Valid values are: Restart, Continue, and FromFile.

PageNumberValue

If Restart is specified for PageProperty, defines the number to use as the starting value (use numeric values only regardless of the number format).

PageNumberFormat

If Restart is specified for PageProperty, defines the format for that number and all subsequent numbers of this type that follow. Valid values are: Numeric, LCRoman, UCRoman, LCAlpha, UCAlpha, Kanji, Zenkaku, UCZenkaku, LCZenkaku, Kazu, Daiji, FullWidthNumeric, UCFull-WidthAlpha, LCFullWidthAlpha, ChineseNumeric, and Text.

ParagraphProperty

Correlates to the options on the **Paragraph** tab of the Numbering Properties dialog. Valid values are: Restart, Continue, and FromFile.

FootnoteProperty

Correlates to the options on the **Footnote** tab of the Numbering Properties dialog. Valid values are: Restart, Continue, UseSame, and FromFile.

FootnoteNumberValue

If Restart is specified for FootnoteProperty, defines the number to use as the starting value (use numeric values only regardless of the number format).

FootnoteNumberFormat

If Restart is specified for FootnoteProperty, defines the format for that number and all subsequent numbers of this type that follow. Valid values are: Numeric, LCRoman, UCRoman, LCAlpha, UCAlpha, Kanji, Zenkaku, UCZenkaku, LCZenkaku, Kazu, Daiji, FullWidthNumeric, UCFullWidthAlpha, LCFullWidthAlpha, ChineseNumeric, and Custom.

FootnoteNumberCustom

If Custom is specified in FootnoteNumberFormat, defines the custom value to use.

TableFootnoteProperty

Correlates to the options on the **Table Footnote** tab of the Numbering Properties dialog. Valid values are: Format and FromFile.

TableFootnoteNumberFormat

If Format is specified for TableFootnoteProperty, defines the format for that number and all subsequent numbers of this type that follow. Valid values are: Numeric, LCRoman, UCRoman, LCAlpha, UCAlpha, Kanji, Zenkaku, UCZenkaku, LCZenkaku, Kazu, Daiji, FullWidthNumeric, UCFullWidthAlpha, LCFullWidthAlpha, ChineseNumeric, and Custom.

TableFootnoteNumberCustom

If Custom is specified in TableFootnoteNumberFormat, defines the custom value to use.

PageStartSide

Correlates to the **1st Page Side** options in the Pagination dialog. Valid values are: FromFile, Next, Left, and Right.

PageDoubleSided

Correlates to the **Pagination** options in the Pagination dialog. Valid values are: 1 (double-sided) and 0 (single-sided).

PageRounding

Correlates to the **Before Saving & Printing** options in the Pagination dialog. Valid values are: DeleteEmpty, MakeEven, MakeOdd, and NoChange.

GeneratedFile-<*mapelemtype*> Section

A GeneratedFile-<*mapelemtype*> section should be included for each generated file to be included in the book.

ComponentType

Specifies the type of generated file to create for this map element. Valid values are:

- Toc - Table of contents
- ListFigure - List of figures
- ListTable - List of tables
- ListPara - Elements and paragraphs
- ListParaAlpha - Elements and paragraphs (alphabetical)
- ListMarker - List of markers
- ListMarkerAlpha - List of markers (alphabetical)
- ListReferences - List of references
- ListFormat - List of formats
- IndexStandard - Standard index
- IndexAuthor - Index of authors
- IndexSubject - Index of subjects
- IndexMarker - Index of markers
- IndexReferences - Index of references
- IndexFormats - Index of formats

NumTags

Specifies the number of “tags” (paragraph style names or marker types) to be included in this generated file. For each tag provide a numbered entry starting with 1 and extending to the value specified in NumTags.

IncludeFiles Section

An IncludeFiles section is used to include FM binary (*.fm*) files in a generated book.

Entries in the IncludeFiles section specify the position and file name of the included file, where the file name is relative to the book file. The key specifies the position and the value specifies the file name. This section can contain multiple entries to add multiple files, just make sure that the “position” values are always unique, and the files are ordered from the lowest position to the highest.

IncludeFileTypes Section

An IncludeFileTypes section is optionally included (if the IncludeFiles section is used) to define the “mapelemtype” values that are associated with the files listed in the IncludeFiles section.

Entries in the IncludeFileTypes section assign the specified mapelemtype values to the files defined by the “position” as used in the IncludeFiles section. The key specifies the position, and the value is the mapelemtype string. You can use values that match those that are map element types in DITA (such as “chapter” or “appendix”) and you can create your own values (such as “titlepage”). Be sure to define the NumberingFirst and NumberingDefault sections as needed to assign numbering and pagination values to these mapelemtype values.

BookBuildOverrides Section

A BookBuildOverrides section is available to override default book-build settings defined in the Book Build Options dialog.

Most entries in the BookBuildOverrides section of the *ditafmx-bookbuild.ini* file mirror those found in the BookBuild section of the *ditafmx.ini* file. If you have questions about the values to be used, refer to the parameters and values in the BookBuild section of the *ditafmx.ini* file.

The following parameters are either not available through the Book Build Options dialog or have features that must be manually entered in the INI file. It’s likely that these options will be included in the UI in a future release.

DitavalName

The value of the DitavalName parameter can be either a registered ditaval name or the path and file name of a ditaval file. If you use the file name option, be sure that this value includes at least one slash. If the ditaval file is in the book output folder, use "./" as the prefix for the file name. Any value that has no slash will be assumed to be a registered ditaval name.

UpdateBook

If set to "1", specifies to run the Update Book command on the generated book. If set to "2", when the book update is performed, the "Apply Master Pages" option is enabled.

RunPostPaginationScript

Similar to the RunScript feature, but instead of running the script before pagination has been completed, use of this parameter runs the script(s) afterwards. Depending on the type of script you're using, you will want to use one or the other (or both).

PostPaginationScriptName

Identical in syntax to the RunScriptName feature.

PostPaginationScriptArgs

Identical in syntax to the RunScriptArgs feature.

MoveFigId

If set to "1", specifies that the @id attribute values are moved to the respective "title" elements when the "Move fig/title" option is enabled. For figures, the @id attribute value is moved to the <fm-figuretitle> element (if one exists in the EDD). This is required to support <fm-xref> references to figure titles. If you do not want these @id values to be moved, set this parameter to "0" (it is set to "1" by default).

MoveTableId

If set to "1", specifies that the @id attribute values are moved to the respective "title" elements when the "Move table/title" option is enabled. For tables, the @id attribute value is moved to the <fm-tabletitle> element. This is required to support <fm-xref> references to table titles. If you do not want these @id values to be moved, set this parameter to "0" (it is set to "1" by default).

BreakToInline

If set to "1", supports the indenting of images in indented content blocks. This parameter must be set manually in a book-build INI file, and defaults to "0" (off). For more information, see Support for Indented Images.

NormalizeConditions

If set to "1", runs a "normalization" process on the condition formats in all book components (including generated lists and any included binary files). This process ensures that the condition settings in all files are the same. The properties of all like-named conditions will be set to the properties of the first instance of that condition in the book. This parameter must be set manually in a book-build INI file, and defaults to “0” (off).

AssignMasterPagesFromMarkers

If set to “1”, enables the **Assign master pages from markers** feature. The text of the first “fmx-masterpage” marker found on a page defines the master page that is applied to that page. This provides an alternative to the StructMasterPageMaps feature in FrameMaker. For additional information, see Custom Master Pages.

RetagElements

This feature attempts to overcome an elusive FrameMaker formatting bug, this setting lets you specify a space-delimited list of element names that are “retagged” at the end of the book-build process. This seems to correct the formatting of paragraphs that contain multiple inline elements.

To enable this option, add a RetagElements entry to the BookBuildOverrides section. The value for this entry is a space-delimited list of elements to retag. This problem seems to occur most in <li> and <note> elements, but may happen elsewhere.

FilterByAttribute

In some cases, the filtering provided by the Filter By Attribute command (FM10 and later) may be preferable to that currently provided by the DITA-FMx mapping of ditavals to conditions. To make use of this filtering option use the FilterByAttribute setting in the BookBuildOverrides section of the book-build INI file.

Set the value of the FilterByAttribute entry to *ATTR-EXPRESSION*>*CONDITION-NAME*. Where *ATTR-EXPRESSION* is the attribute build expression (must exist in the template), and *CONDITION-NAME* is the name of the condition to apply (also must exist in the template).

When the FilterByAttribute entry is enabled, ditaval filtering is not performed, even if specified.

For additional information, see “Filter by Attribute” in Filtering Content.

BuildImagemapHotspots

If you use <imagemap> elements to define linked regions in an <image>, enable the BuildImagemapHotspots parameter to have those linked regions automatically generated by the book-build process. Set this parameter to “1” to enable and “0” to disable (“0” is the default).

The <imagemap> element allows for multiple <area> elements, each of which defines a “hot” region. The <area> element contains a <shape>, <coords>, and <xref> elements. The <shape> element defines the shape of the region, currently only the “rect” (rectangle) shape is supported. The

<coords> element defines the coordinates for the shape. Because only "rect" is supported, the coordinates value is "*x1*,*y1*,*x2*,*y2*" (upper left corner and lower right corner). The <xref> element defines the destination for the hot spot.

The <coords> values are assumed to be in pixels (at 72 DPI). In order to ensure alignment of the coordinates with those of the image itself, you must make use of the "fmdpi" feature (unless your image DPI is 72).

FixTableWidths

This option corrects table widths after master pages have been applied. This is needed if the text column width applied by the master pages differs from that in the base template. Set this parameter to "1" to enable and "0" to disable ("0" is the default).

ShowStatus

Set this parameter to "1" to run the Show Status command on all files in the generated book. Useful for creating deliverables to be used for review purposes ("0" is the default).

HazardToTable

Set this parameter to "1" to enable hazard statement processing in the book build. ("0" is the default.)

HazardSymbolMaxWidth

This parameter provides deliverable-specific control over the hazard symbol width. If not set, the value in the Authoring Options dialog is used. Note that this is a "max" width setting; it does not increase the width of a smaller image.

UseColorFromTable

Set this parameter to "0" to prevent color from being applied to full-width hazard tables in the book build. ("1" is the default, if hazard statement processing is enabled.)

RELATED INFORMATION:

"Book Build Settings" on page 57
"Setting up Book Builds (PDF)" on page 63

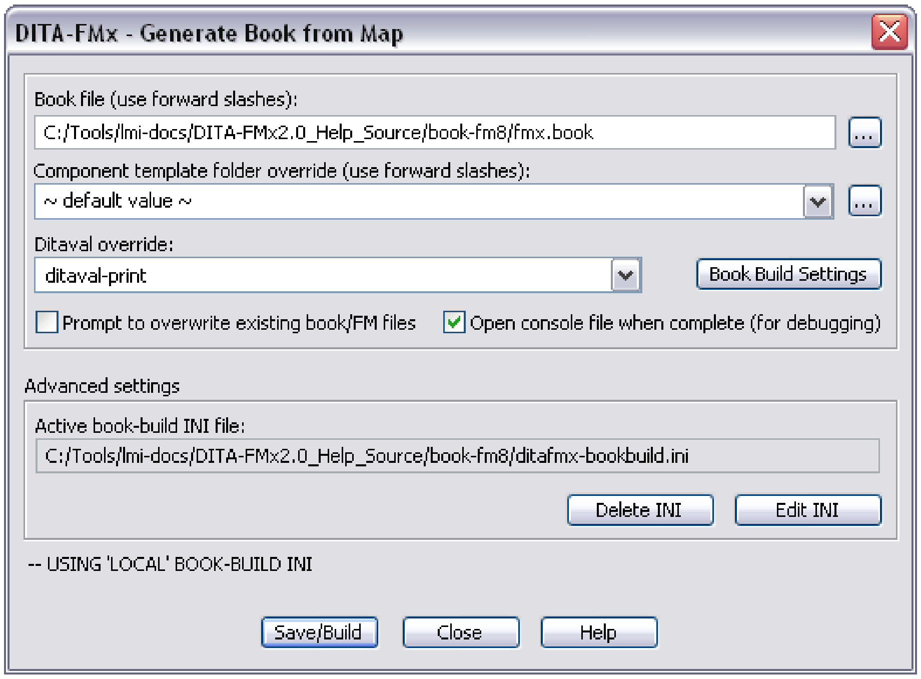
Generate Book from Map

Generates a FrameMaker book and chapter files from the current DITA map.

This command builds a FrameMaker book by creating FM binary files from each top-level topic reference in a DITA map. Any topic references within a <frontmatter> or <backmatter> wrapper element are considered “top-level” topic references and become separate FM files. Additionally, if your bookmap uses a <part> element to organize chapters, the file referenced by the <part> will become a separate FM file, and each child topic-referencing element in the <part> will be considered a top-level topic reference and will be added to the book after the generated part file.

If the current DITA map file has not been saved, it will be saved before processing begins. This command displays the **Generate Map from Book** dialog, which lets you specify the name and location of the book to be generated as well as the ability to select a ditaval file and set an alternate component template folder. You can also edit the associated book-build INI file (*ditafmx-bookbuild.ini*) or modify the default build settings from this dialog.

TIP: *The only required setting in this dialog is the **Book File** name. If you want a quick test of the build process, just specify the book file location and the book and component FM files will be created there. A common practice is to create a folder named “book” within the DITA project folder.*


DITA-FMx - Generate Book from Map

Book file (use forward slashes): C:/Tools/lmi-docs/DITA-FMx2.0_Help_Source/book-fm8/fmx.book

Component template folder override (use forward slashes): ~ default value ~

Ditaval override: ditaval-print

[ ] Prompt to overwrite existing book/FM files [✓] Open console file when complete (for debugging)

Advanced settings

Active book-build INI file: C:/Tools/lmi-docs/DITA-FMx2.0_Help_Source/book-fm8/ditafmx-bookbuild.ini

-- USING 'LOCAL' BOOK-BUILD INI

Figure 3-26: Generate Book from Map dialog

If you don't want to be prompted when overwriting files from a previous build, deselect the **Prompt to overwrite** option and if you want the build log (console file) to automatically open after the build completes, select the **Open console file** option. If the path specified for the book file does not exist, it will be created for you.

The **Component Template Folder Override** option lets you specify an alternate location from which to read the component templates for the build. Similarly, the **Ditaval Override** option lets you select an alternate ditaval file. The default for both of these settings is defined in the book-build INI file (either local or default). The default ditaval can also be set in the Book Build Settings dialog (although the value in the book-build INI will override that in Settings).

Once the book file name has been set, the **Edit INI** button opens the "local" book-build INI file which provides advanced options for controlling details of the book component creation. If no book-build INI file exists in the specified book output folder, the **Edit INI** button copies the default book-build INI to this location before opening it for editing. The **Delete INI** button is provided to remove the local INI file. If no book-build INI file exists in the book output folder, the settings are read from the default location or from the **Book Build Settings** dialog (the settings in the book-build INI override those in the Settings dialog).

If the **Use Language Templates** option is enabled (in the Options dialog), a language-specific book-build INI file is read (if one exists). The language is determined by the @xml:lang value in the map. For example, if the @xml:lang value is "de-de", the book-build INI file name would be *ditafmx-book-build.de-de.ini*.

The book build process starts by aggregating the DITA topic files referenced by the map and any submaps into a single XML file which is imported into FrameMaker using the "Book" structure application which creates the book and chapter files. The resulting FM files are named based on the root topic file's relative path and filename with a ".fm" file extension.

If your map contains key definitions, a keyspace will be generated based on the selected ditaval file (if any). The default keyspace is reset after the build process completes.

Before using this command you should select the desired options in the Book Build Settings dialog, or carefully set up the book-build INI file for this project. The options specified in the Book Build Settings dialog are used for all book builds unless overridden by settings in a book-build INI file. This INI file is read from the same folder as the generated book file or in the user's DITA-FMx folder (**DITA-FMx > Open DITA-FMx Folder**).

Many of the more advanced options, such as numbering and pagination, must be set manually in the book-build INI file. You can maintain separate book-build INI files for each project, or one common INI in the user's DITA-FMx folder.

Conrefs and xrefs that reference content within the book, are updated to point to the new FM files. Any references to DITA files that are not part of the final book, remain pointing at the source DITA file (with the href attribute modified to suit the new path if it has changed). The paths to referenced graphics are updated to account for any changes due to relative differences between the source files and the location of the generated FM files. If your files do reference files that are not part of the book (typically graphics and possibly conrefs), it is important to choose an appropriate location to generate the book so that the FM files and referenced files can be easily moved as needed.

NOTE: *It is not uncommon to get "XML Parser Messages" in the "XML Read Report Log" regarding IDs that have been "already used." This happens if you've used a conref multiple times and that conref contains an ID that is of type "UniqueID." Frame doesn't like multiple instances of the same UniqueID in the same document. DITA doesn't care since they are in different topics. If you get this message, just choose the "OK" button and proceed.*

Maintaining a generated book

One possible way to build and maintain a custom book is described below. The use of a *ditafmx-bookbuild.ini* file and component templates should obviate the need to maintain a book in the way, but this method is offered as an alternative.

1) Generate the book and FM files with the Generate Book from Map command. Think of this as a "throw-away" book, and name it something temporary like *temp.book*.
2) Create a new book by doing a SaveAs from the throw-away book. This is your "real" book, give it an appropriate name.
3) Add your TOC, Index, and other non-DITA-based files to the "real" book.
4) Setup the properties for each file in the "real" book. Select each file and right-click, then choose Numbering, Pagination, or Set Up, as appropriate.
5) Now choose **Edit** > **Update Book** to generate your TOC and Index.
6) Save the "real" book.

Now, when you update your DITA files, re-run the Generate Book from Map command and overwrite the "throw-away" book and the FM content files will be replaced with fresh versions. Just close the "throw-away" book file that's been generated, then open your "real" book and do an **Edit** > **Update** to refresh the TOC and Index and your book is ready to print.

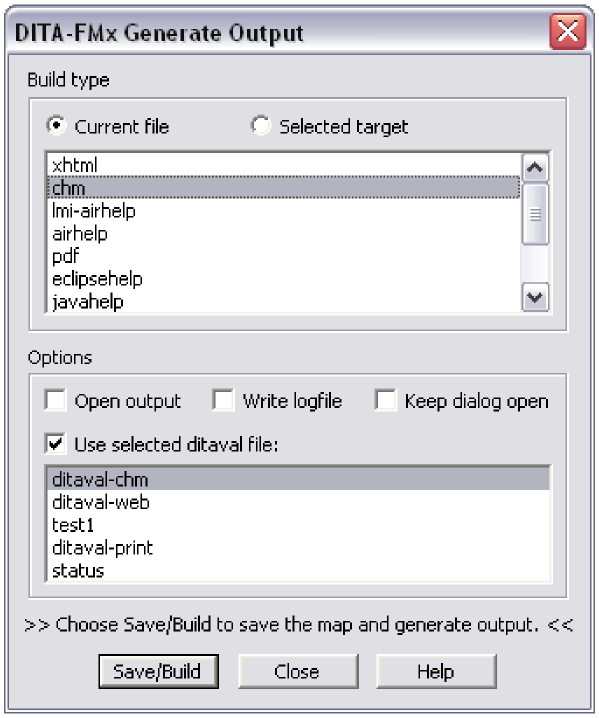
RELATED INFORMATION:

"Setting up Book Builds (PDF)" on page 63
"Ditaval Manager" on page 18
"Apply Ditaval" on page 4
"Setting Up Filtering Groups" on page 32
"Book Build Settings" on page 57
"Book-Build INI file" on page 62

Generate Output

Provides methods for generating output through the DITA Open Toolkit (DITA-OT) from within FrameMaker.

Select the build type and the desired options, then choose the Build button to generate output through the DITA Open Toolkit. If you want to generate output based on the current file (a topic or map) using one of the predefined targets (such as CHM, XHTML, or PDF), use the Current File option. If you have set up an Ant script to build a specific project (not necessarily the current file), use the Selected Target option (OT 1.x only).


DITA-FMx Generate Output

Build type

[✓ Current file] Selected target

xhtml
chm
lmi-airhelp
airhelp
pdf
eclipsehelp
javahelp

Options

☐ Open output ☐ Write logfile ☐ Keep dialog open

☑ Use selected ditaval file:

ditaval-chm
ditaval-web
test1
ditaval-print
status

>> Choose Save/Build to save the map and generate output. <<

Figure 3-27: DITA-FMx Generate Output dialog box

Current File

This option allows you to select from the predefined build targets available for your DITA-OT installation You can also use the Use Selected Ditaval File option to specify filtering.

For setup information, see Generate Output: Current File Option (OT 2.x or later) or Generate Output: Selected Target Option (OT 1.x only).

When this build type is used, a folder is created in the current file's folder named *build-<filename>*. Within that folder, another folder is created to match the selected output type. The files are generated in that folder.

Selected Target

This option lets you run an Ant script and target that you have provided. For information on using this feature, see the section Generate Output: Selected Target Option (OT 1.x only).

Additional options are available for use with either Generate Output build option.

Open Output

If selected, the output folder will be opened after the build completes.

Write Logfile

If selected, a log file is generated and opened after the build completes. If this option is not selected, the build status displays in a command shell window.

Keep Dialog Open

If selected, the Generate Output dialog remains open so you can easily run another build.

Use Selected Ditaval File

This option is only available for use with the Current File output type. If selected, the selected ditaval file is used for the build. Note that you must have added ditaval files with the Ditaval Manager in order for ditaval files to be listed here.

RELATED INFORMATION:

"Publishing with the DITA Open Toolkit" on page 40
"Ditaval Manager" on page 18
"Apply Ditaval" on page 4
"Setting Up Filtering Groups" on page 32
"External Application Settings" on page 56

A Extending DITA-FMx

Provides methods for controlling DITA-FMx from other applications.

DITA-FMx provides a limited set of "CallClient" function calls. These functions can be used by other FDK plugin clients as well as by ExtendScript (in FM10 and later), FrameScript and FrameAC.

NOTE: *Most of these functions are enabled only when the "FMx-Auto" addon is installed. For information on this DITA-FMx addon, please contact Leximation.*

A typical syntax statement shows the call from an FDK client using the F_ApiCallClient() function. A similar call can be made from ExtendScript or FrameScript using the CallClient() function.

All calls require the DITA-FMx client name ("ditafmx"), which must be lower case, and a call client function name (which is not case sensitive). Some calls may have additional (required or optional) parameters. When additional parameters are provided, they are passed as a single tab or vertical bar delimited string. If this string exceeds 512 characters it is truncated.

For example, the FixBookRefs call can make use of the bookId. To pass the bookId (an integer value) convert it to a string and concatenate that string to the initial parameter. For example, if the bookId is 3489237, the FDK call would be as follows:

```
F_ApiCallClient("ditafmx", "FixBookRefs\t3489237");
```

You can also use the vertical bar character as the argument delimiter for calls to the ditafmx client:

```
F_ApiCallClient("ditafmx", "FixBookRefs|3489237");
```

If you would like to have access to specific DITA-FMx functionality that is not currently exposed, please let us know.

RELATED INFORMATION:

"Using DITA-FMx" on page 1
"Installation and Setup" on page 1
"DITA-FMx Commands" on page 1

ApplyDitaval

Runs the Apply Ditaval command to filter the current book or file.

Syntax

```
F_ApiCallClient("ditafmx", "ApplyDitaval |docOrBookIdOrName
|ditavalName [|doClear [|filterType]]");
```

docOrBookId

Id or path and file name of document or book to process. You must use double backslashes as the directory delimiter when specifying the book or document file name. Required.

ditavalName

Name of the ditaval file as shown in the DITA-FMx Ditaval Manager. Required.

doClear

Indicates if existing conditions should be cleared or left alone. This value is ignored for a *filterType* of “2” (filter by deletion). Use “1” to clear or “0” to persist any existing conditions. Optional; default is 1 (clear).

filterType

Specifies the type of filtering to perform. Use “1” for filtering with conditions or “2” for filtering by deletion. Optional; default is 1 (filtering with conditions).

NOTE: *This DITA-FMx API function is only available when the “FMx-Auto” addon is installed. For information on this DITA-FMx addon, please contact Leximation.*

Return Value

0

Failure. Unable to process the specified document or book. It is also possible that DITA-FMx is not available for calls. Verify that DITA-FMx is registered in the *maker.ini* file using the client name of “ditafmx”.

<0

Failure.

>0

Success. Value returned represents the number of conditions applied.

AssignIdToElem

Runs the Assign ID to Element command on the selected or specified element.

Syntax

```
F_ApiCallClient("ditafmx", "AssignIdToElem [|docId |elemId]");
```

docId

FrameMaker ID of the document that contains the specified element. If not provided, the selected element in the current document is used. Optional.

elemId

FrameMaker ID of the element to assign the @id. If not provided, the selected element in the current document is used. Optional.

NOTE: *This DITA-FMx API function is only available when the "FMx-Auto" addon is installed. For information on this DITA-FMx addon, please contact Leximation.*

Return Value

0

Failure. Unable to assign the ID to the specified element. It is also possible that DITA-FMx is not available for calls. Verify that DITA-FMx is registered in the *maker.ini* file using the client name of "ditafmx".

<0

Failure.

>0

Success. Value returned represents the FrameMaker ID of the specified element.

FixBookRefs

Normalizes all references in the files of a FM book file.

Use this function to rebuild and correct all references (xrefs, links, conrefs, and image references) after aggregating loose DITA topics into “chapter” files. Call this function using the “FixBookRefs” parameter to update the active book, or pass the *bookIdOrName* value (as a string) after this parameter to update a specific book. In order to properly update the references in the book, the fm-ditabook/@href attribute must be set to the map’s path and file name. You can set this attribute before calling FixBookRefs, or you can pass this value as the *mapFile* parameter.

Syntax

```
F_ApiCallClient("ditafmx", "FixBookRefs [[|bookIdOrName]
|mapFile]");
```

bookIdOrName

The id or path and file name of the book to process. The book must be open. You must use double backslashes as the directory delimiter when specifying the book file name. Optional; if not provided the “active” book is used, must be provided if *mapFile* is used.

mapFile

The path and file name of the DITA map from which the book was generated. You must use double backslashes as the directory delimiter when specifying the DITA map file name. Optional; if not provided the fm-ditabook/@href attribute must be set to the value of the map’s path and file name before calling FixBookRefs. If this parameter is used, the *bookIdOrName* parameter is required.

NOTE: *This DITA-FMx API function is only available when the “FMx-Auto” addon is installed. For information on this DITA-FMx addon, please contact Leximation.*

Return Value

0

Failure. Unable to update the specified book. It is also possible that DITA-FMx is not available for calls. Verify that DITA-FMx is registered in the *maker.ini* file using the client name of “ditafmx”.

-1

Failure. *bookIdOrName* is not a structured book.

-2

Failure. No non-generated files in book.

>0

Successfully updated the specified book (this is the id of the book processed).

FMxType

Determines the DITA-FMx application type (auto or single user).

Use this call to determine DITA-FMx application type so you'll know which API calls are available.

Syntax

```
F_ApiCallClient("ditafmx", "FMxType");
```

NOTE: *This DITA-FMx API function is available with all versions of DITA-FMx.*

Return Value

0

DITA-FMx is not available for calls. Verify that DITA-FMx is registered in the *maker.ini* file using the client name of "ditafmx".

>0

Indicates the DITA-FMx application type. A return value of 1 indicates the single-user version, and a value of 2 indicates the "auto" version.

FMxVer

Determines if DITA-FMx is initialized and accessible to external calls.

Use this call to determine that the DITA-FMx client is available in addition to ensuring that you are working with the version that you expect.

Syntax

```
F_ApiCallClient("ditafmx", "FMxVer");
```

NOTE: *This DITA-FMx API function is available with all versions of DITA-FMx.*

Return Value

0
: DITA-FMx is not available for calls. Verify that DITA-FMx is registered in the *maker.ini* file using the client name of “ditafmx”.

>0
: Indicates the DITA-FMx version number. A return value of 10113 indicates version 1.1.13, and 20000 indicates version 2.0.00.

InstallApps

Installs structure applications from a ZIP archive.

Syntax

```
F_ApiCallClient("ditafmx", "InstallApps |appname
[|appname...]");
```

appname
: The name of the structure application to install. When installing multiple applications, each is passed as a separate argument separated by the vertical bar character.

Usage

This API installs structure applications using the “stub” file method used by the default DITA-FMx applications. The applications to be installed must be provided in a ZIP archive and are installed to a “root” directory. The “root” directory is defined by the ROOT entry (see below) or is the *FMHOME\Structure\XML* folder if the ROOT entry is not provided. The ZIP file is copied to root folder and extracted into that location. The ZIP archive must be set up with the necessary internal folder structure to be extracted in this way. After the archive is extracted, the stub file is added into the structure application definitions file as a text inset. If the structure application definitions file already contains an application definition of the name being installed, the existing definition will be deleted.

Structure applications installed with this API must first be registered in the AppInstall section of the *ditafmx.ini* file. Each application requires the following entries ("ZIP" and "STUB" entries are required, while "ROOT" is optional).

appname*-ZIP=*path-to-zip

Where *appname* is the structure application name and *path-to-zip* specifies the ZIP archive that contains the application files. This must be an absolute path or relative to the "root" location.

appname*-STUB=*path-to-stubfile

Where *appname* is the structure application name and *path-to-stubfile* specifies the relative path to the installed "stub" file that is to be inserted into the structure application definition file. This path is relative to the "root" location.

appname*-ROOT=*root-path

(Optional) Where *appname* is the structure application name and *root-path* specifies the directory the application is copied to. If this is not provided, the "root" location is the *FMHOME\Structure\XML* folder.

Example *ditafmx.ini* file settings (not using a ROOT entry):

```
[AppInstall]
DITA-MyApp-Topic-1.2-ZIP=DITA-MyApp\DITA-MyApp_1.2_apps.zip
DITA-MyApp-Topic-1.2-STUB=DITA-MyApp_1.2\Topic\structapps-stub_topic_1.2.fm
DITA-MyApp-Map-1.2-ZIP=DITA-MyApp\DITA-MyApp_1.2_apps.zip
DITA-MyApp-Map-1.2-STUB=DITA-MyApp_1.2\Map\structapps-stub_map_1.2.fm
DITA-MyApp-Book-1.2-ZIP=DITA-MyApp\DITA-MyApp_1.2_apps.zip
DITA-MyApp-Book-1.2-STUB=DITA-MyApp_1.2\Book\structapps-stub_book_1.2.fm
```

In this example, all three applications are stored in the same ZIP archive. You could specify separate files if needed. This ZIP archive is copied to the *FrameMaker\DITA-MyApp* folder before the API is used to install the applications.

NOTE: *This API and ditafmx.ini structure is also used by the Configurator command which provides a method for deploying custom structure applications and custom option settings.*

Return Value

0
: Failure. Unable to process the specified document or book. It is also possible that DITA-FMx is not available for calls. Verify that DITA-FMx is registered in the *maker.ini* file using the client name of “ditafmx”.

<0
: Failure.

>0
: Success. Value returned represents the number of applications installed.

RELATED INFORMATION:

"Configurator" on page 16

LoadReferences

Updates the references in the specified or current file.

Use this function to update all references (xrefs, links, conrefs, and image references) in the specified file. Call this function using the “LoadReferences” parameter to update the active document, or pass the *docIdOrName* value (as a string) after this parameter to update a specific document.

Syntax

```
F_ApiCallClient("ditafmx", "LoadReferences [|docIdOrName]");
```

docIdOrName
: The id or path and file name of the document to process. The document must be open. You must use double backslashes as the directory delimiter when specifying the document file name. Optional; if not provided the “active” document is used.

NOTE: *This DITA-FMx API function is only available when the “FMx-Auto” addon is installed. For information on this DITA-FMx addon, please contact Leximation.*

Return Value

0
: Failure. Unable to update the specified document. It is also possible that DITA-FMx is not available for calls. Verify that DITA-FMx is registered in the *maker.ini* file using the client name of “ditafmx”.

-1
: Failure. Invalid *docIdOrName*.

-2
: Failure. document is not a DITA file.

>0
: Successfully updated the specified document (this is the value of the *docId* processed).

MapToWorkbook

Runs the Build Workbook from Map command to create a book file that contains all XML files referenced by a map.

Syntax

```
F_ApiCallClient("ditafmx", "MapToWorkbook |mapName
[|outBookName]");
```

mapName

The full path and file name of the DITA map to be processed. This file must already be open in FrameMaker. The *mapName* value may be “.” (dot) to indicate the “current” file. Required.

outBookName

Full path and file name of the book to be generated. Optional. If not provided the new book name will be *<mapName>.work.book*.

NOTE: *This DITA-FMx API function is only available when the “FMx-Auto” addon is installed. For information on this DITA-FMx addon, please contact Leximation.*

Return Value

0
: Failure. Unable to process the specified document or book. It is also possible that DITA-FMx is not available for calls. Verify that DITA-FMx is registered in the *maker.ini* file using the client name of “ditafmx”.

<0
: Failure.

>0
: Success. Value returned represents the number of conditions applied.

MapToBook

Runs the DITA map to FM book conversion process.

This function generates an FM book and FM chapter files from the specified DITA map. It also runs the FixBookRefs and LoadReferences functions to properly resolve all references. If this function is used to generate an FM book, you should not use the FixBookRefs and LoadReferences functions. All parameters listed below are required.

Syntax

```
F_ApiCallClient("ditafmx", "MapToBook |mapName |appName
|outBookName");
```

mapName

Full path and filename of the DITA map to use as the source for the book. To use the “current” document as the map, use a “.” (period) as the *mapName* value. Required.

appName

Name of the structure application to use for the conversion. Required.

outBookName

Path and filename of the new book file to generate. If a relative path is used for the *outBookName*, it is assumed to be relative to the DITA map being processed. Required.

NOTE: *This DITA-FMx API function is only available when the "FMx-Auto" addon is installed. For information on this DITA-FMx addon, please contact Leximation.*

Return Value

0
: Failure. Unable to update the specified document. It is also possible that DITA-FMx is not available for calls. Verify that DITA-FMx is registered in the *maker.ini* file using the client name of "ditafmx".

<0
: Failure. The number of parameters passed (as a negative number). If more or fewer than 4 parameters are passed, that number will returned as a negative number.

>0
: Successfully converted the specified map (this is the value of the generated *bookId*).

OpenDITA

Opens a file as a DITA file, using the proper structure application.

Syntax

```
F_ApiCallClient("ditafmx", "OpenDITA |ditaFileName");
```

ditaFileName
: Path and file name of DITA file to open. You must use double backslashes as the directory delimiter when specifying the DITA file name. Required.

NOTE: *This DITA-FMx API function is available with all versions of DITA-FMx.*

Return Value

0
: Failure. Unable to open the specified file. It is also possible that DITA-FMx is not available for calls. Verify that DITA-FMx is registered in the *maker.ini* file using the client name of “ditafmx”.

<0
: Failure.

>0
: Success. Value returned represents the docId of the opened file.

XrefToHyperlink

Runs the Xref to Hyperlink command.

Syntax

```
F_ApiCallClient("ditafmx", "XrefToHyperlink
[|docOrBookIdOrName]");
```

docOrBookIdOrName
: Id or path and file name of document or book to process. You must use double backslashes as the directory delimiter when specifying the book or document file name. Optional, if not provided, the “active” document or book will be used.

NOTE: *This DITA-FMx API function is only available when the “FMx-Auto” addon is installed. For information on this DITA-FMx addon, please contact Leximation.*

Return Value

0
: Failure. Unable to process the specified document or book. It is also possible that DITA-FMx is not available for calls. Verify that DITA-FMx is registered in the *maker.ini* file using the client name of “ditafmx”.

-1
: Failure.

>0

Success. Value returned represents the number of xrefs hyperlinked.

B DITA-FMx Glossary

Terms specific to DITA-FMx.

RELATED INFORMATION:

"Using DITA-FMx" on page 1
"DITA-FMx Commands" on page 1

book-build INI file. An INI format file (*ditafmx-bookbuild.ini*) used to store the setup and processing information required to produce a consistently formatted FrameMaker book from a DITA map using the DITA-FMx **Generate Book from Map** command.

The **Generate Book from Map** command looks for a book-build INI file in the book output folder, then in the user's DITA-FMx folder (**DITA-FMx > Open DITA-FMx Folder**).

book-build INI file (ditafmx-bookbuild.ini)

RELATED INFORMATION:

"The Book Build Settings Dialog and the Book-Build INI File" on page 65
"Book-Build INI file" on page 69
"Generate Book from Map" on page 83

component template. Structured FrameMaker template files applied to the book components after the initial Book structured application template has been applied to the DITA map to generate the book and component files.

There are two types of component templates, layout templates (with a *tpl~* prefix) and generated list templates (with a *gentpl~* prefix). Layout templates are applied to book components based on their map element type (<chapter>, <appendix>, etc.). Generated list templates are applied to components that are children of the <booklists> element (in <frontmatter> and <backmatter>), in order to make use of the default FrameMaker generated list feature.

Fundamentally, the layout component templates are typically a renamed copy of the Book template, with modifications made to alter the page layout or formatting definitions. The underlying structural model must match that of the

Book template, but the context rules may differ in order to achieve the desired effect.

Component templates are referenced by the BookTemplatesDir parameter in a book-build INI file. This parameter specifes a folder that contains the component templates available for a particular build.

RELATED INFORMATION:

"Description of the Structure Application Files" on page 16
"Generate Book from Map" on page 83

DITA Open Toolkit. The DITA Open Toolkit, or DITA-OT for short, is a set of Java-based, open source tools that provide processing for DITA maps and topic content.

You can download the OT and install it for free on your computer to get started with topic-based writing and publishing (dita-ot.github.io).

DITA Open Toolkit (DITA-OT)

- DITA-OT
- DITA-OTK
- OT

element template. An element template is used to provide boilerplate (default) content for a new DITA topic created with the **New DITA File** command. One or more element templates may be created for each topic type, and added to a folder (specified in the **New File Options** dialog). When creating a new file you can select from the available element templates for each topic type.

RELATED INFORMATION:

"Creating Element Templates" on page 49
"New DITA File" on page 2

key element reference. A key element reference refers to content that is defined within the <topicmeta> of a <keydef> element. This is not an industry-defined term, but one created to provide a name in which to refer to this construct since it does differ from other types of referenced content in DITA.

RELATED INFORMATION:

"Insert Key Element Reference" on page 29

C Revision History

Describes the changes between versions of DITA-FMx.

RELATED INFORMATION:

"DITA-FMx Commands" on page 1

2.0.08 - 12 October 2020

New features

Added support for FM 2020

Only offering support for the 64-bit version of FM 2020 at this time.

Added DynamicTargets option for DITA-OT publishing

For use with DITA-OT 3.2 or later. This default option will automatically display the publishing targets for the currently installed OT plugins in the Current File list of the Generate Output dialog. To disable, set Dynamic-Targets=0 in the BuildFile section of the *ditafmx.ini* file.

Structure application updates

Common EDD inset files

- None

Topic app (topic_1.2.edd.fm, topic_1.2.template.fm)

- None

Book app (book_1.2.edd.fm, book_1.2.template.fm)

- None

Book XSLT (bookmap2fmbook.xsl)

- None

Component template updates

- None

Element template updates

- None

Bug fixes / minor updates

Changes to the DITA-OT options in the External Application Settings dialog

- When selecting OT versions prior to 1.6 as well as 3.1.2 and 3.1.3, you will be prompted to enter the DITA-OT version.
- For OT versions 2.0 or later, the Environment Setup File is automatically set to the *bin/dita.bat* file in the selected DITA-OT directory. For earlier versions the Environment Setup File must be manually selected.

Keyref resolution issue

- A bug was causing the keyspace file to become corrupt, thus preventing it from loading, and preventing keyrefs to resolve properly. This has been fixed.

ditafmx.ini updates

- BuildFile/DynamicTargets=[<1>/0]

2.0.07 - 18 April 2019

New features

Added support for FM 2019

Only offering support for the 64-bit version of FM 2019 at this time.

Installer now updates maker.ini

The installer application now provides the option to comment out the default “fm-dita” API client entries in the *maker.ini* file. You no longer need to do this manually when installing DITA-FMx for the first time.

Added support for image formatting overrides

Allows the use of many of FrameMaker’s anchored frame properties. See Support for Indented Images, for details.

Added file name filtering in Insert Image dialog

The Insert Image dialog now provides the option for substring filtering of the file name to insert, making it much easier to locate image files in a long list.

New Configurator command

Simplifies the setting of default DITA-FMx and FrameMaker options by changing values in the *ditafmx.ini* and *maker.ini* files. Useful for helping to ensure team members are using the same settings or for switching between different settings for different projects. See Configurator, for details.

Structure application updates

Common EDD inset files

- Updated *commonElements.edd.fm* to add formatting support for note/@type=notice and note/@type=warning. Added context rules to apply note.notice, note.warning, note.entry.notice, and note.entry.warning para tags.

Topic app (topic_1.2.edd.fm, topic_1.2.template.fm)

- Added new para tags to Topic template for note type support added to EDD (note.notice, note.warning, note.entry.notice, and note.entry.warning).
- Updated commonElements EDD inset in Topic EDD.

Book app (book_1.2.edd.fm, book_1.2.template.fm)

- Added new para tags to Book template for note type support added to EDD (note.notice, note.warning, note.entry.notice, and note.entry.warning).
- Updated commonElements EDD inset in Book EDD.

Book XSLT (bookmap2fmbook.xsl)

- None

Component template updates

- Import updates from EDD and template (for note/@type=notice and note/@type=warning) into component templates.

Element template updates

- Updated *new~reference~simple ref.fm* element template to remove hard-coded @class attribute values (causing validation issues from the lack of trailing space).

Bug fixes / minor updates

DITA-OT 2.x updates

- Added new *ditafmx.ini* parameters for the BuildFile section to provide additional support for DITA OT 2.x (or later) builds:

  BuildFile/OT2Params=<user params> (defaults to nothing)

  BuildFile/OT2Params-<build>=<user params> (applies only to current build, overrides OT2Params if set)

  BuildFile/OT2Logging=[<Verbose>/Debug/None] (None = errors only, if not set, defaults to Verbose)

- If the OT2Params value specifies an output location (via '-output=' or '-o='), that will override the creation of the default builds directory based on the current filename.

Auto-prolog updates and fixes

- Added optional *ditafmx.ini* parameter PrologOptions/UnknownDateStr for setting critdate/@created if unknown. (default “0000-00-00”, was “UNK”)
- On file open, if critdate/@created is "UNK" it is changed to the value specified by PrologOptions/UnknownDateStr (default "0000-00-00").
- If “New Author” option is not selected in the Auto-Prolog Options dialog, that value is no longer added to the prolog (as you’d expect).

New building blocks

- Added $TITLE_SPACETODASH and $TITLE_SPACETODASHLC. Converts the document title to a dash-delimited string.

Image element enhancements

- Added support for the image/@scale attribute. If the @height and @width attributes are empty, the value of the @scale attribute will be used for scaling the image.

  TIP: *When inserting an image with the Insert Image dialog, enter NN% in the DPI field to set the @scale attribute.*

- Added (limited) support for @scalefit.
- Support editing of @height, @width, @scale, @scalefit via attribute window.

Image element fixes

- When referencing images, the drive letter is now treated in a case-insensitive manner, so an absolute path isn't created to the same drive.

Xref fixes

- If an xref with alt text wraps to a new line in the XML, the space (new line) between wrapped words is no longer lost on file save.
- Added a feature to automatically add alt text to xrefs based on the target element type when selected in the Reference Manager. In the *ditafmx.ini* file, the INIOnly/AutoXrefTextElems parameter specifies a space delimited list of elements. If not set, no automatic alt text is added.

Conref fixes

- When an @id is required on an element, if that element is included in a conref structure when saving the minimum required elements as a placeholder in the XML, a temporary @id is now added to that element. (Prevents Xerces error messages on file save.)

Keyref fixes

- Improved the resolution of nested keyrefs.
- Improved keyref resolution in titles and topicrefs.

Conkeyref fixes

- A conkeyref of a table that has a row which contains no cells, will render correctly now.

Updated InstallApps API

- Added ROOT option, allowing installation of custom apps in locations other than the default “FMHOME” area.

Fixed indexterm seealso forced sort string duplication

- Added support for multiline see/seealso entries (happens when entry has multiple words).
- While authoring, forced sort coding for seealso entries is not added (authors should not add); coding is added during the book-build process.

Reporting of read-only files

- To disable to message box generated when opening read-only files set the *ditafmx.ini* file INIOnly/ReportRO parameter to "0".

Processing instructions

- Fixed round-tripping of comments and unstructured marker PIs.

Book-build INI updates

- None.

ditafmx.ini updates

- INIOnly/AutoXrefTextElems
- INIOnly/ReportRO
- BuildFile/OT2Logging
- BuildFile/OT2Params
- PrologOptions/UnknownDateStr

2.0.06 - 20 November 2017

New features

Added support for FM 2017

IMPORTANT: *For installations on FM 2017, the update #2 (14.0.2) is required for DITA-FMx due to updated FDK library.*

Added support for Lightweight DITA

- Includes new LwDITA-FMx-Topic, LwDITA-FMx-Map and LwDITA-FMx-Book structure apps (based on latest pre-release LwDITA DTDs).

- Auto-detects “xdita” in topic/@domains to “do the right thing” with the LwDITA model.
- New Map adds “title” as /map/topicref/navtitle (structapp applies formatting to that navtitle).
- Wrap a <simpletable> with a <fig>, fig/@outputclass is set to 'fm:tablewrapper' (allows the <title> in the <fig> to be formatted as a table title).

Added support for DITA-OT 2.x and 3.x

The Generate Output command now detects the DITA-OT version selected in the External Application Settings dialog and uses the proper syntax for the OT command line. The transformation type (format) entries in the BuildFile section of the *ditafmx.ini* are set to those available in OT 2.5.4; if you require alternate transformation types, you’ll need to edit that section.

Added support for hazardstatement elements

Provides support for rendering <hazardstatement> elements for authoring as well as new publishing options. For details, refer to Working with Hazard Statements.

fmx:maxlen feature (beta)

New fmx:maxlen feature provides interactive length checking of an element. Enable by adding add the attribute “fmx:maxlen” to an element definition. Specify a default value of the maximum length in characters (make it read only and possibly hidden). This attribute should not be written to XML. This attribute will be used if set in the *ditafmx.ini* with INIOnly/ElemMaxLenCheck=1.

xref-container feature (beta)

New element/feature to allow markup (such as images) within an xref.

Structure application updates

Lightweight DITA structure applcations

- Three new structure applications are provided to support Lightweight DITA. LwDITA-FMx-Topic, LwDITA-FMx-Map, and LwDITA-FMx-Book can be installed by extracting the contents of the *LwDITA-FMx_apps.zip* file found in the *DITA-FMx-2* installation folder.

Topic app (topic_1.2.edd.fm, topic_1.2.template.fm)

- Added support for <fm-var> in the <shortdesc> general rule.

- Added support for <hazardstatement> processing, including updates to the hazardstatement (and related element) context rules and the addition of numerous hazard* paragraph tags.

Book app (book_1.2.edd.fm, book_1.2.template.fm)

- Added support for <fm-var> in the <shortdesc> general rule.
- Added support for <hazardstatement> processing, including updates to the hazardstatement (and related element) context rules, the addition of numerous hazard* paragraph tags, a new Hazard table format, and new hazard* color definitions.

Book XSLT (bookmap2fmbook.xsl)

- Updated to support SAXON as the XSLT processor (the default in FM2015 and FM 2017). Both the XALAN and SAXON versions of this XSLT file are provided in the Book application folder. FM 2015 and FM2017 users should be able to use this out-of-the-box, but installing on FM versions before FM 2015, you'll need to copy the XALAN version for use. For details, see the *_SAXON-or-XALAN.txt*. file in the Book application folder.

Component template updates

- All updates to the Book application EDD and template are reflected in the component templates.
- Added new *tpl~topicref.fm* component template.
- Added new *tpl~topicref-HAZALT.fm* component template that provides an example of the alternate hazardstatement table publishing option.

Bug fixes / minor updates

Table formatting

- Tables with titles are now properly indented during the book-build process (if the "Indent Tables" option is enabled).
- Table widths (for both simpletables and regular tables) are now properly sized in the book-build process.
- If the "Preserve Table Widths" option is enabled, the percentage values are not assigned to tables where @pgwide='1' (they remain full page width).
- Fixed an error with table width sizing on tables where @pgwide is greater than "1".

Image updates

- Inserting an <image> into an existing <fig> now sets the @placement attribute to “break”.
- Default value in EDD for image/@placement attribute is now used as the default value in the Insert Image dialog.

Graphic overlay object issues

- To support use in LwDITA (where data/@datatype is not valid), graphic overlay objects will now use data/@name instead. This migration will happen on file save and should be transparent to users. If you want to continue using @datatype rather than @name for this purpose, set the following parameter in the *ditafmx.ini* file .. INIOnly/UseDatatype=1
- Added full support for FP_NumPoints (number of points in a polyline). This was being written but not read.

Ditaval issues

- The *by deletion* option now supports filtering with a missing @attr or @val attribute. This has been supported in the *as conditions* option.

Conditional filtering update

- A new feature was added for those who want manually apply conditional tagging to content in your DITA files, and have those conditions controlled (hide/show state) along with the conditionalization settings in the Options dialog. The hide/show state of conditions with names based on the default “DITA-*” conditions and ending with “-Set” (i.e. “DITA-Prolog-Set”) are toggled along with the default conditions.

Reference Manager issues

- The values presented in the conrefend field should now be correct regardless of other selected values.
- The Element Data list populates with only values valid at the current insertion point.

File reference issues

- File references (@href attributes) that use encoded spaces (“%20”) now load properly.
- UpdateReferences command now works properly when the auto-load options are off.

Key referencing issues

- An unresolved @keyref on an image no longer deletes that image element on file open.
- Updating an existing @conkeyref (with the Reference Manager) no longer deletes that element.
- Saving a file with an unresolved @conkeyref, no longer writes that node with a missing element name.
- Inline conkeyrefs are no longer converted into their parent elements on file save.
- Inline elements within key element references are no longer converted to visible markup, now elements are unwrapped (content remains, but markup is gone). This is not an ideal solution; will eventually fully support inline elements.

Topicref issues

- Topicref navtitles are now truncated at 250 bytes in length. When content is truncated, the navtitle will end with an ellipsis. To change this length set the following *ditafmx.ini* parameter accordingly .. INIOnly/NavtitleLen
- Updating a topicref (with the Update References command) which references a topic with multibyte characters will no longer create an empty navtitle element.
- Fixed the handling of navtitles to allow for formatting that applies a prefix.

Auto-prolog issues

- Auto-adding “revised date” with the “new author” option not selected, now works as expected.

Book-build INI updates

- Added BookBuildOverrides/FixTableWidths=1 setting to correct table widths after master pages are applied. This is needed if the new master page(s) are wider than the base page width.
- Added new BookBuildOverrides/HazardToTable setting to enable/disable the hazardstatement processing in a book build.
- Added new BookBuildOverrides/UseColorFromTable=0 setting to prevent color from being applied to full-width hazard tables.
- Added new BookBuildOverrides/HazardSymbolMaxWidth setting to provide deliverable-specific control over the hazard symbol width. If not set, the value in the Authoring Options dialog is used.

Note that this is a “max” width setting; it does not increase the width of a smaller image.

- Fixed error in handling PDFDocInfo settings. If an item was sent to an empty value, then entire process would fail.

ditafmx.ini updates

- In a fresh installation (or after deleting the INI), the new *ditafmx.ini* file is written as UTF-8.
- Checking for MaxRefLevels parameter is done first in the INIOnly section then in the General section. (Logic was reversed before, but this parameter is “ini only”.)
- Added new “beta” INIOnly/ElemMaxLenCheck=1 setting to enable the fmx:maxlen feature (described above).
- The BuildFile/OTVer parameter is now a “no-op”. The OT version is now read from the *<OTDIR>\lib\configuration.properties* file. If this file doesn’t exist, the OT version is assumed to be 1.5 (that file didn’t exist before OT 1.6).

CMS support

- Updates to fix conref and conrefend problems with xDocsFMx.

Authorization

- Maintenance renewal message now allows 360 days to buy renewal.

2.0.05 - 13 September 2016

New features

Added support for new number formats

Added support for these number formats: FV_FN_NUM_-FULL_WIDTH FV_FN_NUM_FULL_WIDTH_UC FV_FN_NUM_-FULL_WIDTH_LC FV_FN_NUM_CHINESE_NUMERIC

Added support for these *NumberFormat values in bookbuild INI: Kanji, Zenkaku, UCZenkaku, LCZenkaku, Kazu, Daiji, FullWidthNumeric, UCFullWidthAlpha, LCFullWidthAlpha, ChineseNumeric (last 4 only valid for FM9 and later).

New labels for external topicrefs

Added prefix to topicref labels in map. For external hrefs, use protocol as prefix (only happens when "Title" is selected in options).

Improved support for indexterm ranges

Added support for start/end range Index marker syntax. Sets @start and @end values based on marker text.

The text of the Index marker (including subentry text) is the value for the @start and @end attributes, but once those attributes are set, you'll have to manually change the attributes if you want to change the values. (Changing the marker text won't change the attribute values.)

The first time an Index marker is converted into an indexterm, it'll "do the right thing" and split on semicolons, even if the marker has multiple start/end range markup. But, if you edit an existing marker that has a @start or @end attribute, and you tack on a semicolon with a new entry, it'll split those, but the new entry will have the same @start or @end value. This basically duplicates the attributes on the element. This may or may not be what you really want, but something to be aware of.

At this time, indexterms are not supported in maps (thus ranges are not either). The content of the "end" indexterm is ignored.

Structure application updates

Map EDD (map_1.2.edd.fm)

Limited support for topichead/glossref (visually in map, still not properly supported by XSLT though).

Topic app (topic_1.2.edd.fm, topic_1.2.template.fm)

Added limited support for hazardstatement elements.

Book app (book_1.2.edd.fm, book_1.2.template.fm)

Added limited support for hazardstatement elements.

Book XSLT (bookmap2fmbook.xsl)

Provide minimal support for topichead, it is now no longer deleted, unwrapped instead, @navtitle is lost but remaining markup persists.

Bug fixes / minor updates

Table cell rotation

If a table cell is rotated, and you use the reference page coloring/properties feature, the rotation specified in the topic now takes precidence over any rotation defined in the reference page.

Also, now multiple values are supported in @outputclass (space-delimited).

indexterm fixes

Fixed case where a trailing semicolon and space was creating empty indexterm.

Fixed case where empty Index marker was writing <fm-indexterm> to DITA, causing error.

keyrefs deleted on file open

Fixed problem with XDocsFMx (and possibly elsewhere) with keyrefs that got deleted on file open.

keydef with angle brackets

A keydef with content that uses angle brackets will display properly now.

Rendering of invalid conkeyref

An invalid conkeyref now renders as "[CONKEY:<conkeyval>]".

xref relink fix

Relinking an xref to a new topic type, now updates @type to the new topic type.

Reference Manager fix

Reference Manager now properly scrolls to the selected element and element data.

Navtitle not lost if target topicref is missing

On map save, if the referenced topic is missing, no longer writes “FILE NOT FOUND” text to navtitle.

fm-var no longer deletes conditional tagging

A fm-var in the first child element of a block element (like abstract/shortdesc or ul/li), no longer causes conditional tagging on the block element to get lost.

Crash on edit of invalid attribute

If an attribute is copied to an element where it is invalid, editing that attribute with the Set Attributes command no longer crashes Frame.

image handling fixes

The file save operation no longer writes an empty @align attribute for images.

A mis-cased image file extension no longer causes image reimport for book builds.

Insert image dialog

Changed the Insert Image dialog so the default alignment is “sticky” (persistent in session) if no default is defined in the EDD, otherwise the default is based on the EDD default.

Change to book build default

Changed the default for book-build PromptToOverwrite to "Off".

Book-build messages

Eliminated the “ERROR: Unable to open ditaval file! []” message during book builds with reltables.

Auto-prolog support for required abstract

The auto-prolog feature will now work properly if your EDD specifies that the abstract is required..

Fixed disabling of keyboard-only commands

The keyboard-only commands were disabled after switching to XML code view and back. This has been fixed.

Where Used topicref fix

Now properly reports topicref references in the selected map.

2.0.04 - 13 September 2015

IMPORTANT: The FM 13.0.1 update changed the default XSLT processor from XALAN to SAXON which causes the DITA-FMx Book application’s XSLT script to fail. More information and the fix for this is described in the topic, Special Instructions for FM12 and Earlier.

New features

Added support for FM 2015

DITA-FMx now supports FrameMaker 2015.

New PDF Doc Info book-build command

Added PdfDocInfo feature to the book-build INI file (ini-only). Specify the PDF metadata values to add into the generated book. Added new PdfDocInfo section to the default book-build INI file.

Refer to PdfDocInfo Section in the “Book-Build INI file” topic for usage details.

New Show Status command and book-build option

Added Show Status command (**Utilities > Show @status**) to apply special status-oriented conditions to the current document. The default conditions will render content marked with @status=’new’ or @status=’changed’ with change bars. You can customize the template to apply custom formatting for all @status states.

Added ShowStatus BookBuildOverrides param (ini-only). Set this parameter to “1” to run the Show Status command on all files in the generated book. Useful for creating deliverables to be used for review purposes.

New variable building block

Added <$VAR(name)> building block variable.

Structure application updates

Abbreviate domain Elements EDD (abbreviatedomain.edd)

Modified the character tag associated with the following elements: abbreviated-form

Programming domain Elements EDD (programmingdomain.edd)

Added an associated character tag for the following elements: apiname

Software domain Elements EDD (softwaredomain.edd)

Added an associated character tag for the following elements: cmdname, msgnum

UI domain Elements EDD (uidomain.edd)

Modified the character tag associated with the following elements: shortcut, wintitle

Common Elements EDD (commonElements.edd.fm)

Added support for figure title levels (just an example) for BreakToInline on fig title before image (title element definition, firstParaRules context rule).

Modified or added the character tag associated with the following elements: boolean, keyword, state, term, tm

Topic DTD (fmx-ditabase_1.2.dtd)

Added definition for &nbsp; entity (apparently to be deprecated in DITA 2.0).

Topic EDD (topic_1.2.edd.fm)

Replaced “Red” color in “Unsupported” format rule with “DITA_Unsupported”.

Reimport updated shared EDDs.

Topic template (topic_1.2.template.fm)

Added conditions to support ShowStatus feature (DITA-StatusNew, DITA-StatusChanged, DITA-StatusUnchanged, DITA-StatusDeleted)

Added/updated the following character tags: term.abbrev, prog.api.name, boolean, soft.cmd.name, keyword, soft.msgnum, state, ui.shortcut, term, tm, ui.wintitle.

Imported updated EDD.

Map DTD (fmx-map_1.2.dtd)

Added definition for &nbsp; entity (apparently to be deprecated in DITA 2.0).

Map EDD (map_1.2.edd.fm)

Replaced “Red” color in “Unsupported” format rule with “DITA_Unsupported”.

Added choice values for permissions/@view attribute.

Map template (map_1.2.template.fm)

Added “DITA_Unsupported” color.

Imported updated EDD.

Book DTD (fmx-book_1.2.dtd)

Added fm-ditabook attributes (metadata mappings) for pubinfo-org, pubinfo-completedyear, pubinfo-completedmonth, pubinfo-completedday, permissions-view, and category.

Added definition for &nbsp; entity (apparently to be deprecated in DITA 2.0).

Book EDD (book_1.2.edd.fm)

Added fm-ditabook attributes (metadata mappings) for pubinfo-org, pubinfo-completedyear, pubinfo-completedmonth, pubinfo-completedday, permissions-view, and category.

Replaced “Red” color in “Unsupported” format rule with “DITA_Unsupported”.

Reimport updated shared EDDs.

Book template (book_1.2.template.fm)

Added conditions to support ShowStatus feature (DITA-StatusNew, DITA-StatusChanged, DITA-StatusUnchanged, DITA-StatusDeleted)

Added/updated the following character tags: term.abbrev, prog.api.name, boolean, soft.cmd.name, keyword, soft.msgnum, state, ui.shortcut, term, tm, ui.wintitle.

Imported updated EDD.

Book XSLT (bookmap2fmbook.xsl)

Added metadata mappings for pubinfo-org, pubinfo-completedyear, pubinfo-completedmonth, pubinfo-completedday, permissions-view, and category. XML mappings are as follows:

```
//publisherinformation/organization -> pubinfo-org
//publisherinformation/published/completed/year ->
pubinfo-completedyear
//publisherinformation/published/completed/month ->
pubinfo-completedmonth
//publisherinformation/published/completed/day ->
pubinfo-completedday
//permissions/@view -> permissions-view
//category -> category
```

Added support for author-defined conditions. Conditions created in the topic XML files can be set up to persist in the generated FM files. Add othermeta element(s) to the map where @name='author-condition' and @content='CONDITION-NAME'. By default these conditions are created with the color Black, but if you want specific styling, add the conditions to the template or component template.

Changed the variable named 'node' to 'nodename'.

Deleted unused 'fmtext' variable definition.

Book component templates (tpl~*.fm and gentpl~*.fm)

Rebuilt all component templates from the updated Book template.

Imported new variables and conditions into generated list templates.

New Book stub file for FM 2015 (structapps-stub_book_1.2_13.0.1.fm)

With the 13.0.1 update, the default XSLT processor is now SAXON. The XSLT import script in the Book app requires XALAN, so this new stub file includes the new Processor node to set the XSLT processor to XALAN.

For new installations, this stub file will be used by default, but for upgrades, it won't, and the Generate Book from Map command will likely fail. For details, see Special Instructions for FM12 and Earlier.

Bug fixes / minor updates

FrameMaker XML Author alerts

Added alerts that display when running DITA-FMx under FM XML Author. Instead of generating errors when using commands that don't work under XML Author, you'll now see messages.

book-build INI updates

Added the General/AttrAsCondDefaultState parameter to the default book-build INI file.

ditafm.ini update modification

Modified the test for determining when to update the *ditafm.ini* file (yes, the default FM *ditafm.ini* not *ditafmx.ini*) on FM10 and later. This fixes an edge case problem that results in the inability to open topic files after installation.

http-based hrefs are not "normalized"

URLs in an external xref are no longer "normalized," which caused errors.

fm-var and fm-data-marker updates

Fixed bug with fm-var and fm-data-marker creation for FM12. These elements moved to the start of the container paragraph, but now stay where created.

An fm-var within the source of a conref, no longer causes the conref to get wrapped in an fm-var element.

fm-var elements recreated from text in a file are no longer truncated when that text wraps in the XML.

When hiding area/data/data-about, markers (fm-data-marker elements) are no longer hidden.

data to fm-data-marker and fm-var processing

Fixed data to fm-data-marker and fm-var processing so the 'topic/data' class isn't passed on to the new element, allowing it to take on the default class of the EDD (fmx/fm-data-marker or fmx/fm-var).

Long ids with long path no longer crash FM

Fixed crasher with very long ID values caused the path to exceed an internal limit.

Empty indexterms no longer crash FM

Empty indexterms will now properly round-trip from XML to FM and back.

Relative paths in book-build INI

Relative paths are now valid in the book-build INI for Pre/Post-Build-Script.

Ditaval filtering of reltables

Fixed ditaval filtering of reltables. Now handles attributes with multiple values; also properly checks for include vs. exclude, so include wins.

External xref updates

Fixed external xrefs so the link text sticks (doesn't revert to match the @href).

Fixed external xrefs so they don't grow characters (when "Element Banner Text" is enabled).

When creating Hypertext markers for external xrefs, if target is PDF, make into proper syntax (openlink) unless it's a URL

imagemap fixes

Fixed problems with inserting imagemaps, shape, coords, xref didn't render values properly.

book-build updates

Disable "Element Banner Text" display when running a book-build (FM11 and later).

BreakToInline figure handling

Updated BreakToInline processing so it works for figs where the title isn't moved.

IndentTables handling

Updated IndentTables processing so when checking parent element's text object, if the parent and the table anchor's object don't have the same FP_Placement (Format: "In Column", "Run In Head...", etc.), the table is left unindented.

Keyspace Manager update

Fix to INI-reading code which truncated keyspaces listed in Keyspace Manager.

fm-xref updates

fm-xrefs referencing another file now properly add the value for @type attribute (target element name).

Fixed fm-xrefs so you can click to edit existing fm-xrefs.

Stopword handling

When removing stopwords, if the result of processing leaves an empty string, set it to the first “word” in the string.

Element template support

Updated new file from element template so any existing IDs in the element template are updated (if Auto-Add IDs is enabled).

New file command update

When creating a new file of the same name that exists, it will now successfully delete the existing file (after prompting for approval).

2.0.03 - 31 March 2015

New features

Added support for ditaval filtering by deletion

This feature more closely emulates the ditaval filtering done by the DITA-OT. Instead of content being hidden by FrameMaker’s conditional text feature, the content is deleted from the generated FM files. Additionally, if the root element of a generated file is to be excluded, that FM file is deleted from the book.

To support this feature, new INI parameters have been added. In the *ditafmx.ini* file you’ll see `BookBuild/DitavalFilterType` and in the book-build INI you can now use `BookBuildOverrides/DitavalFilterType`. The values for both of these can be 0 (no filtering), 1 (filter by conditions), or 2 (filter by deletion).

Another new *ditafmx.ini* parameter is `INIOnly/FilteringAttributes`. This specifies the attributes that are considered when performing ditaval filtering. The default value is a space-delimited string of attribute names “`audience product platform otherprops rev props`”.

The **Apply Ditaval As Conditions** command has been renamed to **Utilities** > **Apply Ditaval** and now offers both filtering with conditions and filtering by deletion options.

Added support for ditaval filtering of reltables

Ditaval filtering during the book-build process will now affect related links added from relationship tables, based on filtering attributes applied to <relrow> and <topicref> elements.

Book-build enhancements

The generated book file is now saved automatically at the end of a book-build.

The following options have been added to the book-build INI:

- BookBuildOverrides/MoveFigTitles - Move fig/title after image (2) or to the end of the fig (1).
- BookBuildOverrides/DitavalFilterType - (described above)
- DitavalDefaultsOverrides section - replicates values from the DitavalDefaults section in the *ditafmx.ini* file. If used in book-build INI file, overrides those settings for the current book build. Only applies to ditaval filtering that uses the filter by conditions option.

New command: Insert Topicrefs

The Insert Topicrefs command lets you insert multiple topicrefs at one time. This is only available on FM10 and later.

Enhanced support for Japanese indexes

The <sort-as> element now works properly for Japanese indexes.

Added new parameter to the *ditafmx.ini* file to support the Japanese index marker name. On Japanese systems the Index marker is not named "Index." Set the `IndexOptions/IndexMarkerType` parameter to the correct name.

Added new option "Conditionalize required-cleanup"

This option is similar to the other "conditionalize" options, and allows the automatic conditionalizing of any <required-cleanup> elements. It applies a DITA-Cleanup condition to the <required-cleanup> elements.

Added new API command: MAPTOWORKBOOK

If you're performing automation with DITA-FMx, the new MAPTOWORKBOOK command will allow you to generate a "work book" from a map. This API is only available if FMx-Auto is enabled.

Added support for the AZARDI EPUB reader

[Experimental] If you want to use an EPUB for online Help in DITA-FMx, you can do that with the AZARDI reader. To enable this, add the following entry to the *ditafmx.ini* file:

```
ExternalApplications/EPUBReaderAppExe=<path-to-e
xe>
```

Structure application updates

Map DTD (fmx-map_1.2.dtd)

Updated to add proper general rule for indexterms in maps.

Topic EDD (topic_1.2.edd.fm)

Updated <synph> element context rules to check for special values for @outputclass attribute. If @outputclass is set to “element”, the <synph> content is wrapped in angle brackets, and if set to “attribute” an “@” prefix is added.

Topic template (topic_1.2.template.fm)

Added DITA-Cleanup condition.

Imported updated EDD.

Topic rules (topic_1.2.rules.txt)

Added ‘row’ and ‘entry’ rules.

Book XSLT (bookmap2fmbook.xsl)

Added new metadata to fmx-variable mappings for: bookpartno, booknumber, maintainer/person, and maintainer/organization.

Book EDD (book_1.2.edd.fm)

Added new fm-ditabook attributes for metadata to fmx-variable mappings: bookpartno, booknumber, maintainer/person, and maintainer/organization.

Added support for <fm-figuredesc> (element definition and modified <fig> general rule).

Updated <synph> element context rules to check for special values for @outputclass attribute. If @outputclass is set to “element”, the <synph> content is wrapped in angle brackets, and if set to “attribute” an “@” prefix is added.

Book DTD (fmx-book_1.2.dtd)

Added new fm-ditabook attributes for metadata to fmx-variable mappings: bookpartno, booknumber, maintainer/person, and maintainer/organization.

Book template (book_1.2.template.fm)

Added DITA-Cleanup condition.

Imported updated EDD.

Normalized the main Book template with the “gentpl” component templates to ensure that all para tags exist in all templates. This makes the

"Save As PDF" from a generated book work better so the para tags are listed properly by default.

Book rules (book_1.2.rules.txt)

Added 'row' and 'entry' rules.

Book component templates (tpl~*.fm and gentpl~*.fm)

Rebuilt all component templates from the updated Book template.

Bug fixes / minor updates

Book-build conditional tagging cleanup

Removed the process that added the DITA-Topicmeta conditon to generated chapter files. Also made sure that all setting of conditions in generated chapter files was also applied to the generated list files. This seems to eliminate the book-build errors reported about inconsistent conditions.

Generate Workbook from Map updates

The Generate Workbook from Map command has been updated to significantly decrease the time required to generate the book. For maps with many (100+) topics, this previously took 20 minutes or more. Now it might take a minute for 500 topics.

Auto-add @id updates

The auto-add @id process now works when wrapping with an element that requires an @id attribute.

Fix Linebreaks option deprecated on FM12

The Fix Linebreaks option is no longer needed on FM12 (in fact it was causing problems), so it is not available on that version of FM.

Linebreak PIs within preformatted elements

Linebreak PIs are no longer added within preformatted elements.

Cleanup on coderef handling

Edge cases of multiple <coderef> elements in a <codeblock>, now work better.

Properly reads nested key maps

It's often useful to create nested key maps, and now DITA-FMx will properly interpret the key definitions in those maps.

2.0.02 - 10 August 2014

New features

Support for XDocsFMx 2.0

This update provides support for the XDocsFMx 2.0 connector providing DITA 1.2 features with the XDocs CMS.

Keyrefs to a topic now use title text, also provides alt text support

When the associated <keydef> for a key element reference (like a <keyword>) does not define <topicmeta> content, the effective element content is now pulled from the topic title referenced by the <keydef> @href. Also, key element references can make use of alternative text when no <topicmeta> content is defined.

Structure application updates

NONE.

Bug fixes / minor updates

Book-build process supports complex paths

The Generate Book from Map command can now handle more complex directory structures than in previous releases. A topicref/@href that starts with a "..\" should resolve properly now.

Conrefs to figs are not deleted when using the Move Figure Title option

Previously, when using the "Move Figure Titles" option for book-builds, a conref to a <fig> would result in that element being deleted from the output. This has been fixed.

Insert Key Element Reference dialog supports multiple keyspaces

Available keys are now properly updated when a different keyspace is selected.

Using keyrefs with images on Windows Vista/7/8

Images that use a keyref with no href fallback should work properly now on Windows Vista/7/8.

Update References: xref option works properly

The Update References xref option no longer deletes xrefs.

Set Attributes dialog provides Edit INI button

A new Edit INI button lets you easily make edits to the *filtergroups.ini* file to control filtering groups displayed in the dialog.

2.0.01 - 9 June 2014

New features

Generate Book from Map enhancements

New options have been added to the Generate Book from Map dialog.

The functionality of the Edit Book-Build INI button has been modified. If a book-build INI file exists in the book output folder, that file is opened for editing. If no INI file exists at that location, the default book-build INI file is copied into that location from the “user’s” DITA-FMx folder or from the DITA-FMx installation folder, then that file is opened for editing. If the Use Language Template option is enabled (in the Options dialog), a language-specific INI file is opened or copied using similar logic. Also, this button is now labeled “Edit INI” and a “Delete INI” button is also provided in order to remove the “local” INI file and revert processing to the default INI.

The Component Templates Override option lets you select a new component templates folder to override that specified in the book-build INI file. If the AltBookTemplatesDirs section exists in the book-build INI, the folders specified in that section are presented as alternate override options in the Component Templates Override list.

A new “Open Console File on Completion” option is available to automatically open the console log file after the build completes. Also, the *build-consfile.txt* file that is created in the build folder now contains only the console log messages for the current build (instead of all previous log messages for the session).

Now, when running the Generate Book from Map command, you are presented with a file browse dialog if the current file is not a map.

Added Apply Master Pages option to Book Build Settings dialog

Apply Master Pages has been added as a sub-option to the Update Book option in the Book Build Settings dialog.

Use Language Templates option

The Use Language Templates option is now available in the Options dialog (previously an INIOnly setting). This setting lets you maintain language-specific book-build INI files and component templates, based on the @xml:lang value in the DITA map.

Added option to strip stop words from new file names

If the New File Name format uses a $TITLE* building block for creating new file names from titles, any “stop words” are stripped from the proposed file names. The INIOnly/StopWords parameter lets you specify a space-delimited list of stop words or a file name that contains the stop words.

Conrefs now partially supported in maps

Use of conrefs in a map title or other metadata element will now be available as a FrameMaker variable if the ImportAttrsAsVars book-build INI option is enabled. This feature should be considered “experimental” for now.

Structure application updates

REMEMBER: *Any modifications to a “common” EDD file must be updated in the structure application EDD (both Topic and Book), then imported into the associated templates as well as any component templates. Similarly, any modifications to templates will require similar changes to component templates. This has been done in the provided files, but if you’re implementing this on your custom files you’ll need to perform this operation.*

Common EDD updates (commonElements.edd.fm)

- Removed prefix label strings from note element definition. These strings are now defined in paragraph tags in template that are applied by the context rules.
- Changed context rules for note to remove LevelRule for note within table cell (entry | stentry | propdesc | chdesc | choption), replaced with specific match on lists within cell. This prevents extra indents within table cell based on context outside of table.

Common Topic EDD updates (topic.edd.fm)

- Removed prefix label string from related-links element definition. This string is now defined in a paragraph tag in template applied by the context rules.

Common Task EDD updates (task.edd.fm)

- Changed step/@id and substep/@id types to UniqueID to allow linking from an <fm-xref>. Use the ParaNumOnly cross-ref format to link to a step or substep number.
- Removed prefix label strings from prereq, result, and postreq element definitions. These strings are now defined in paragraph tags in template that are applied by the context rules.

Book and Topic application template updates

- Added/updated paragraph tags to support new prefix and label strings assigned by EDD modifications described above.

Book DTD updates (fmx-book_1.2.dtd)

- Updated to add specific general rules for certain inline elements that disallow PCDATA (data-about, copyright, critdates, metadata, keywords, prodinfo, vrmlist, menucascade). This fixes a situation that caused odd whitespace problems, in particular with menucascade structures that lost the ">" delimiter.

Bug fixes / minor updates

Coderef Manager update

The browse button in the Coderef Manager starts browsing from the current file's location if no value is specified.

The maximum line length for a coderef file is now 1024 characters (increased from 256).

No @href specified for a key definition

If a key definition didn't specify an @href value, the map name was being used instead, which caused odd problems. This issue has been resolved.

Keyspace is automatically rebuilt for all book builds

If a map contains key definitions, a keyspace is generated for that map to ensure proper key resolution.

Xref to Hyperlink processing cleanup

In certain cases where a topic contained a large number of id attributes, the XrefToHyperlink processing during the book-build process could cause FrameMaker to crash or "sleep" for very extended periods. This appears to be resolved now.

Xref "alt text" works properly

An xref created with alternate link text preserves that link text after reopening the document. Previously, this link text reverted to the target title text.

Book build with images that have invalid file extensions

If an image had a file extension that didn't match its actual file type, FrameMaker could crash during the book-build process. This issue is now resolved.

Move Figure Titles option no longer deletes figure titles

Occasionally if the Move Figure Titles book-build option was enabled, the moved figure titles would actually get deleted. This issue has been resolved.

Generate Book from Map with no map open didn't build related links

Running the Generate Book From Map command with no map open would result in the related links not being added to the generated FM files. This issue is now resolved.

Maps in "deep" paths

The *~mapdata.ini* file contains the build-specific settings and mappings. Maps in very long paths were causing this data to be deleted with each restart of FrameMaker. This issue is now resolved.

Element template updates

If no path is specified for the Element Templates Folder, element templates are detected in an *element-templates* folder within the current structure application folder (the folder that contains the app's template file). To support this, an *element-templates* folder (along with a sample element template) has been added to the Map application folder.

When a map element template is used, the <title> and @id are updated properly.

Where Used update

Because the Where Used command generates a report that is a FM binary file, it cannot operate in Author View mode. Now, when running this command in Author View, you are prompted if it is OK to switch to WYSIWYG View.

Search in Files update

If no folder is specified for the Search Scope, the browse button in the Search in Files dialog now starts browsing from the current file location rather than the file system root.

2.0.00 - 14 April 2014

Changes that may affect content migration

- The FmDpiUseOutputclass *ditafmx.ini* parameter controls the attribute used for the “fmdpi” feature. In DITA-FMx 1.1, this was the @otherprops attribute, but in DITA-FMx 2.0 it now defaults to the @outputclass attribute. If you want to continue using the @otherprops attribute, set this parameter to “0”.
- The FmXrefUseOutputclass *ditafmx.ini* parameter controls the attribute used to store the <fm-xref> (cross-ref) format name. In DITA-FMx 1.1, this defaulted to the @otherprops attribute, but in DITA-FMx 2.0 it now defaults to the @outputclass attribute. If you want to continue using the @otherprops attribute, set this parameter to “0”.
- Opening DITA 1.1 topic files in a DITA 1.2 model (the default application model in DITA-FMx 2.0) will work fine, and you should be able to open those same files back in a DITA 1.1 application model without trouble. However, adding new elements that make use of DITA 1.2 features (such as using keyrefs, conkeyrefs, or coderefs), will prevent you from opening these files using a DITA 1.1 application. If you plan on working with a DITA 1.1 model, just install the structure applications from DITA-FMx 1.1.
- Opening DITA 1.1 map files in a DITA 1.2 model (the default application model in DITA-FMx 2.0) will work fine. However, the DITA 1.2 map model adds the option for a <navtitle> element as a descendant of topicrefs. By default, when working in maps, the <navtitle> will be added only for new topicrefs. In the Map Options dialog you can change this to Never add <navtitle> elements, or to add <navtitle> elements on map open.

New features

Generate Book from Map dialog

A dialog has been added to the Generate Book from Map command. This dialog lets you select the output book file name as well as select a ditaval file to apply to the build. The selected book name persists between builds and remains associated with each DITA map. This dialog also lets you access the Book Build Settings dialog and create/edit the associated book-build INI file.

Keyspace Manager dialog

Lets you specify a root map as the source for a keyspace. The keyspace can optionally be filtered by a ditaval file.

For details, see Keyspace Manager.

Key reference support for xrefs, links and conrefs

The Reference Manager allows you to select a key and a sub-element within the referenced file to specify the keyref for xrefs, links and conrefs.

For details, see Using the Reference Manager.

Conref range support

The Reference Manager lets you specify the end element for a conref range.

For details, see Using the Reference Manager.

Key element reference support

The Insert Key Element Reference command lets you insert an element whose content is defined by content within a key definition.

For details, see Insert Key Element Reference.

Glossary term keyref support

When inserting a key element reference to a glossary entry, by default, the content of the glossary term (<glossterm> element) is included in the <term> element.

If you enable Glossary Term Swapping (in the Book Build Options dialog) and your glossary topic provides a <glossSurfaceForm> element and a <glossAlt> abbreviation, the first instance of the <term> in a chapter will use the <glossSurfaceForm> value and the rest will use the <glossAlt> value.

Insert Image dialog provides keyref support

The Insert Image dialog provides easy access to image properties and referencing methods.

Coderef support

Non-DITA content references can be created with the coderef element. This is ideal for documentation that uses code samples.

For details, see Using coderefs.

FrameMaker variable support has been enhanced

The Prepare Variables and Rebuild Variables commands have been removed and the Auto-Prepare Variables on Save option has been replaced with a new Convert Variables to fm-var Elements option. The new <fm-var> element round-trips to DITA as a <ph> element. This allows you to insert FrameMaker variables at any time and those variables will be rebuilt when the file is opened later.

When opening a file from DITA-FMx 1.1 that used the old ph/@keyref='fmvar:VARNAME' markup, it will convert into the new format.

New auto-prolog options

Added "new author" option to support multiple authors of the same document. Also, now provides support for auto-prolog control in maps.

For details, see Auto-Prolog Options.

Automatically adds author and date to draft-comments

When inserting a draft-comment element, the author and date are added to the element.

Provides control of when navtitles are added to maps

Navtitle elements can be added to a map on file open or only on topicref insertion (or not at all). This is controlled in the Map Options dialog.

Set Attributes dialog is available in three sizes

The DITA-FMx Set Attributes dialog/pod can be set to small, medium, and large depending on your needs. If you have enabled the "filtergroups" feature to make use of attribute value checkboxes, the small dialog supports up to 10 checkboxes, medium supports up to 20, and large supports up to 40. Each checkbox option name will now display up to 24 characters.

Additional support for table formatting and layout

Tables can now make use of the outputclass attribute to assign custom ruling and shading to rows and cells. Also, the new Preserve Table Widths option provides better control over the width of tables, and the Indent Table to Parent Element option allows control over table indents (both are found in the Authoring Options dialog).

For details, see Working with Tables.

Added support for imagemaps

The **Insert Imagemap Data** command (on the Utilities menu) lets you import map data created by a third party application (or by hand) from an XHTML file. This transfers the map data (area, shape, coords, xref) into the DITA <imagemap> element.

The **Test Hotspots** command is keyboard-only and accessed via the shortcut `Esc,T,H`. It draws rectangles on the <imagemap> image to represent the "rect" hot spots.

The BuildImagemapHotspots book-build INI option generates “hot spot” regions in an image based on the <area> data provided by an <imagemap> element. To make use of this option use the BuildImagemapHotspots setting in the BookBuildOverrides section of the book-build INI file. A value of “1” enables the option and a value of “0” disables it.

Structure application updates

New structure applications based on the DITA 1.2 model are provided.

Refer to the *DITA-FMx_1.2\common_info.fm* file in the structure application folder for details.

The structure of the default DITA-FMx EDDs have been modified to support the $attribute syntax in the Initial Table Format context rule. This allows the table object element definitions to apply the format stored in the outputclass attribute, without needing to explicitly list all table formats in the template. For details, see Working with Tables.

NOTE: *A “Learning” application is provided (in the Topic folder), but it is not considered to be production-ready. It is provided for testing purposes only and will be updated in a future release.*

Bug fixes / minor updates

Added the FmDpiUseOutputclass parameter

By default, when using the “fmdpi” feature, DITA-FMx will now use the @outputclass attribute to store the DPI data instead of the @otherprops attribute as used in DITA-FMx 1.1. If you want to continue using the @otherprops attribute, set the FmDpiUseOutputclass parameter to “0”. This change only affects how files are written; any existing images that use @otherprops for this data will still render correctly regardless of the setting.

The FmXrefUseOutputclass parameter now defaults to “1”

By default, when saving an fm-xref, the cross-reference format name is stored in the outputclass attribute instead of the type attribute. This more accurately follows the intent of these attributes.

Allow specification of topic and map ID in New DITA File dialog

When creating a new topic or map, the proposed ID is included in a field that may be edited if you have a requirement for an alternate value.

New File Name Options are now available for maps

When creating a new map, the map file name is automatically generated based on the title, according to the New Map File Name Format option.

Insert File As dialog displays for new files in map

If the insertion point is in a map when creating a new topic file, you are prompted to select the type of element to insert into the map.

Added $T_D and $TOPIC_ID building blocks

The $T_D building block provides a 1 or 2 digit date, and $TOPIC_ID is a replacement for $UNIQUEID.

Added limited support for image/alt text

Image alt text (via the <alt> element as a child of <image>) is available for editing in FrameMaker through the Object Properties (Object Attributes) dialog. The alt text is stored as an object attribute of the image's anchored frame, then written back to a proper <alt> element on file save.

Added browse button to External Xref dialog

To assist in linking to non-DITA content a browse button has been added.

Provide option to customize the New File shortcut keys

Two new INIOnly *ditafmx.ini* settings are available to define custom shortcut keys for the New File menu commands. Refer to the documentation on NewMapShortcuts and NewTopicShortcuts.

Update to the ApplyTemplates book-build feature

The ApplyTemplates book-build option now selects all available formats when importing from the template(s) to the generated files. Previously for FM10 and later, some of the formats were not imported.

Added ShowBuildTimes book-build INI option

This option prints the current build time of each step in the book-build process, which can help in debugging. To enable this option, add a ShowBuildTimes=1 entry to the General section. A value of "1" enables the option and a value of "0" disables it.

Added FilterByAttribute book-build INI option

In some cases, the filtering provided by the Filter By Attribute command (FM10 and later) may be preferable to that currently provided by the DITA-FMx mapping of ditavals to conditions. To make use of this filtering option use the FilterByAttribute setting in the BookBuildOverrides section of the book-build INI file.

Set the value of the FilterByAttribute entry to *ATTR-EXPRESSION>CONDITION-NAME*. Where *ATTR-EXPRESSION* is the attribute build expression (must exist in the template), and *CONDITION-NAME* is the name of the condition to apply (also must exist in the template).

When the FilterByAttribute entry is enabled, ditaval filtering is not performed, even if specified.

For additional information, see “Filter by Attribute” in Filtering Content.

Added RetagElements book-build INI option

In an attempt to overcome an elusive FrameMaker formatting bug, this new setting lets you specify a space-delimited list of element names that are “retagged” at the end of the book-build process. This seems to correct the formatting of paragraphs that contain multiple inline elements.

To enable this option, add a RetagElements entry to the BookBuildOverrides section. The value for this entry is a space-delimited list of elements to retag. This problem seems to occur most in <li> and <note> elements, but may happen elsewhere.

Added PostPaginationScript book-build INI options

The standard Script options run before pagination is complete, which may be what’s needed for some scripts, but if you need to run a script after pagination, use this new option. To enable, add the RunPostPaginationScript, PostPaginationScriptName, and PostPaginationScriptArgs entries to the BookBuildOverrides section.

Global structure applications may be used

Structure applications defined in the global structure application definitions file (FM11 and FM12) will now properly be recognized by DITA-FMx.

Authoring options were moved to a new dialog

In an effort to clean up the Options dialog, the settings that were previously in an “Authoring Options” section of the main dialog are now in a sub-dialog accessed via the Authoring Options button.

Additionally, some settings were migrated from the INIOnly section of the *ditafmx.ini* file to the Authoring Options dialog.

Addressed problems with conref tables and moved table titles

If the “Move table titles” book-build option is enabled, tables that include titles and are included by conref should no longer get deleted.

1.1.18 - 14 January 2014

New Features

Added support for FrameMaker 12

DITA-FMx should work properly with FM12. It does not work fully with FrameMaker XML Author.

Added support for identifying flattened conrefs

Added INIOnly/FlattenedConrefAttr parameter. If set to specify an attribute name, adds *attrname*='flattened-conref' to flattened conrefs. This is useful for a publishing process that needs to identify content that came from a conref.

Added "post pagination" script book-build options

The default book-build script runs before the content has been conditionalized and other processes run which affect pagination. If you have a script that needs to run after pagination, use these new BookBuildOverrides options (RunPostPaginationScript, PostPaginationScriptName, and PostPaginationScriptArgs).

Structure Application Updates

Import XSL script updates

The *expandOrig.xsl* and *bookmap2fmbook.xsl* scripts were updated to add support for references to maps from a backmatter and other elements.

Book template updates

The *book_1.1.template.fm* file was updated to include "title-appx.0" and "title-part.0" paragraph tags which were referenced by the EDD (added in an earlier update).

Bug Fixes / Minor Updates

Conditionalize Data option conditionalizes area now

The Conditionalize Data option now applies the DITA-Data condition to any area elements.

All 20 custom attribute values display properly in Set Attributes dialog

When using all 20 custom attribute values in the *filtergroups.ini* file, and if those strings were long, it was possible that they would not all display completely in the Set Attributes dialog. This has been fixed.

1.1.17 - 25 November 2013

New Features

Added new file name building blocks and modifiers

Added <$TITLE_NOSPACECAMEL> and <$TITLE_NOSPACECAMELLOW> building blocks. Also added 'U', 'L', and 'T' (upper, lower, title case) modifiers.

Added book-build option to modify the variable prefix

If using the General/ImportAttrsAsVars feature to generate FM variables based on map metadata, by default the variables are created with an "fmx-" prefix. If you'd like a different (or no) prefix, set the General/AttrAsVarPrefix to that value.

Added option to preserve table widths

If the *ditafmx.ini* INIOnly/PreserveTableWidths parameter is set to "1" the relative width of each table will be stored in the table/@pgwide attribute. This value will be used to adjust the width of each table when rendering the topic in FrameMaker. For additional details, see INI-Only Settings.

Structure Application Updates

No changes or updates have been made to the structure applications.

Bug Fixes / Minor Updates

Use a custom character format for xrefs

Previously, if you tried to use a character format for xref elements other than the default "link.asis" would result in the xref being non-functional. This has been fixed; you can now specify any character format.

The consfile.txt file is now copied to the book-build folder

At the end of each book-build, the current *consfile.txt* file is copied to the book build folder.

A semicolon in an indexterm no longer causes FM to crash

Previously, if an indexterm contained a semicolon, FrameMaker could crash. This no longer happens.

Multi-level forced sort in an indexterm works properly now

Previously, the forced sort (index-sort-as) mechanism only worked for single level index entries. It now works for multi-level entries.

Indented tables resize properly

Previously, if a table format specified that a table was indented, when resized to fill the text column it would overflow one side. Tables now properly fill the text column regardless of the indent.

Using an element template with a map on FM9-FM11 works properly

Attempting to create a map with an element template on FM9-FM11 was not working. This works properly now.

The “New DITA File” menu items are available in FM11 Author View

The “New DITA File” menu items were previously not functional in the FM11 Author View.

New DITA file titles will be added even if your EDD defines a prefix

Previously, if your EDD added a prefix to the topic title, the New File command would not add the new topic title to the file. This has been fixed.

Items on the New DITA File menu are now alpha-ordered

The new topic types on the New DITA File menu will now appear alphabetically ordered.

http-based image references work properly

Using an http-based reference to images had mixed results. It now works as expected.

Table cells rotate properly

Rotated table cells that contain child elements would previously not remain rotated after reopening a DITA file.

Content within a conreffed table row or cell no longer remains in the DITA

Typically, a conref element is empty (or just contains the minimal structure to make it valid). Previously, DITA-FMx would leave the content of a conreffed row or cell in the DITA file. Now this content is omitted.

Clipboard is no longer lost when using an import XSLT

Previously, when using an import XSLT with the DITA-FMx import client application, the clipboard would become non-functional. This has been fixed.

1.1.16 - 1 December 2012

New Features

Updates to the BookTemplatesDir parameter in the book-build INI file

You can now use the $STRUCTDIR variable in the BookTemplatesDir parameter to specify a path in the FrameMaker Structure directory.

The path specified in the BookTemplatesDir parameter is now always relative to the generated book file. Previously, when the default book-build INI file was used (the one in the DITA-FMx folder), the path was relative to the INI file. Now it's consistent.

Added support for alternate default image placement values

When opening a file that contains image elements with no @placement value, the default was to set those to @placement='break' (even though the default value for @placement is 'inline'). Now you can set the *ditafmx.ini* parameter INIOnly/ImagePlacement to control this for your needs. Valid values are: 0 = use EDD default; 1 = set to 'inline'; 2 = set to 'break' (the default).

Structure Application Updates

Book XSLT (expandOrig.xsl)

Support for topicref/@print='no' was added to expandOrig.xsl.

Bug Fixes / Minor Updates

Whitespace normalization has been updated

If the option is enabled, whitespace normalization is performed when files are opened. There were times when this didn't work properly for some types of pretty-printed files. Now, the following processing is applied: multiple whitespace characters are collapsed into a single whitespace, and all leading and trailing whitespace is removed from each element (unless it is defined as a “preformatted” element).

DITA-FMx commands are now available in the FM11 Author View

FM11 adds Code View and Author View options. The DITA-FMx commands are now available in Author View.

Fixes to the DITA-OT build process including support for OT 1.6.x

The ditafmx-ant.xml file has been updated to support changes made in OT 1.6. Also fixes an error introduced in 1.1.15 to add support for EPUB output through the DITA4Publishers plugin. If the D4P plugin was not installed, the build process would fail. These problems have been addressed.

See-also indexterms round-trip properly

See-also indexterm elements that wrap to another line in the XML would generate an extra nesting level after reopening the file. This no longer happens.

Support for topicref/@print=’no’ is working again

This support was added in 1.1.11, but then lost in 1.1.12 when the XSLTs were updated. Code to support this filtering has been added to the expandOrig.xsl file.

Topicrefs that include a ‘topicid’ reference no longer cause problems in xrefs

Previously a topicref/@href that included the topicid value (as in “filename.xml#topicid”) could, under certain situations result in a non-functioning xref to that topic. This type of @href value is no longer a problem.

Xrefs that include a prefix or suffix are no longer deleted on file open

If an xref element definition specifies a prefix or suffix, that element will properly render on file open and in book-builds.

Running the Apply Ditaval as Conditions command on a book file, works

Previously, running the Apply Ditaval as Conditions command on a book file didn’t not actually apply the conditions. It does now.

New conditions created by the Apply Ditaval as Conditions command are no longer colored red

Previously, when the Apply Ditaval as Conditions command created a new condition (because it didn’t exist), it was colored red. This doesn’t seem to serve any useful purpose, and now no longer happens.

Use an element template when creating a new file on Windows 7 or Vista

The element template feature now works properly on Windows 7 or Vista.

Replacing an existing image updates the @href attribute

You can now select an existing image and use **File** > **Import** > **File** to replace that image without deleting and reinserting the image element.

Create Archive command now creates a properly named archive file

As of DITA-FMx 1.1.15 the Create Archive command started creating an archive file with "more" at the end of the filename, and incorrectly included a list of files in the archive. The archive is now created properly.

Missing "gentpl" files no longer cause a book-build crash

If the BookTemplatesDir parameter in the book-build INI file wasn't referencing a valid location that contained the necessary "gentpl" templates for a bookmap that you're trying to build, the Generate Book from Map command would crash. This no longer happens.

Reset Status command now offers a cancel option

The Reset Status dialog can be dismissed without actually resetting the status attribute values.

1.1.15 - 30 July 2012

New Features

Added support for FrameMaker 11

DITA-FMx can now be installed on FrameMaker 11. No FM11-specific modifications have been made to the code to support new FM11 features.

Added feature to disable "Runaround" on images

The default setting for the Runaround property is "On", which causes problems when placing callouts over images. When using DITA-FMx this will automatically be set to "Off". To disable this feature and revert to the default setting, set the INIOnly/RunaroundOff parameter in the *ditafmx.ini* file to "0".

Strips space that follows open or close element tags

The *ditafmx.ini* file GeneralImport section includes a new parameter, StripTrailing. If set to 1, the import process strips space characters from the XML that follow opening or closing block-level element tags. This only strips the spaces when there are no other non-space characters in the node that follow the open or close element.

Added support for multiple DITA-OT environments

The *ditafmx.ini* file BuildFile section can include new parameters EnvironmentSetup-<buildname> and DitaDir-<buildname>.

Added support for EPUB publishing through the DITA for Publishers plugin

The *ditafmx-ant.xml* file has been updated to include a new "epub-out" target. Install the DITA for Publishers plugin, then update the Ant file in your DITA-OT installation, and add this entry to the BuildFile section in the *ditafmx.ini* file.

Added book-build support for master pages

When generating a FM book using the Generate Book from Map command, if your base template or component-templates make use of a StructMasterPageMaps reference page to control the application of master pages, those master pages will be applied. To enable this feature you must be using a book-build INI file, and in the BookBuildOverrides section set the UpdateBook parameter to "2".

An alternative to the StructMasterPageMaps reference page method for applying master pages is to add an "fmx-masterpage" marker to a page. If this marker is found during the book-build process, the text of the marker specifies the name of the master page to apply to that page. This may be useful for the occasional setting of master pages. The fm-data-marker element can be used to set this marker. Note that in order to use this feature, the "fmx-masterpage" marker must be defined in the Topic and Book templates. For additional information, see Custom Master Pages.

Added support for ditaval filtering using missing @att or @val values

The Apply Ditaval as Conditions command (and book-build ditaval process) now supports processing using prop elements that don't include the att or val attribute values. A ditaval prop that doesn't specify the val attribute will match on all "att" attribute values. A ditaval prop that doesn't specify the att attribute will match on all "val" attribute values.

Added support for ditaval filtering on topicref elements

Through updates to the Book import XSLT, filtering attributes applied to topicref elements are now passed on to the corresponding fm-ditafile elements in the generated FM files. Because of this update, filtering specified in the map will now allow you to hide entire topics in the generated book. Note that this does not apply to the root topicref elements at this time; the content will get filtered, but the file will still exist in the generated book. This could be achieved through further customization of the Book import XSLT if needed.

Added support for structapps.fm from Text Structure Consulting, Inc.

Text Structure Consulting (www.txstruct.com) provides a custom structure application definition file that offers extended features over the default option from Adobe. DITA-FMx now functions properly when this file is in use.

Structure Application Updates

Book DTD (fmxbook.dtd)

The *fmxbook.dtd* file has been updated as follows:

- Added the fm-ditabook/@xml:lang attribute. This resulted in an error when generating a book from a map that included an xml:lang value.
- Added authorname and copyrholder attributes to the fm-ditabook element.
- Added filtering attributes to the fm-ditafile element.

Book EDD (book_1.1.edd.fm)

Added corresponding attributes to support DTD updates:

- Added the fm-ditabook/@xml:lang attribute.
- Added authorname and copyrholder attributes to the fm-ditabook element.
- Added filtering attributes to the fm-ditafile element.

Book Template (book_1.1.template.fm)

The book template was updated as follows:

- Imported updated EDD.
- Added fmx-masterpage marker definition.

Book XSLT (bookmap2fmbook.xsl and expandOrig.xsl)

A number of updates were made to the *bookmap2fmbook.xsl* file.

- Incorrect references to a draftinfo element were changed to draftintro.
- Support was added for a preface element that can reference a ditamap.
- Additional support was added for the migration of map metadata to book-level attributes.

- Updated processing to migrate filtering attribute values from topicrefs to the associated fm-ditafile element in the generated FM files. This allows ditaval/conditional filtering based on map-level attributes.

It is possible that these updates could cause problems for existing book-build processing, so the old XSL files are also provided as *bookmap2fmbook.1.1.14.xsl* and *expandOrig.1.1.14.xsl*.

Book Component Templates

The updated EDD was imported into the following component templates: *tpl~appendix.fm*, *tpl~chapter.fm*, *tpl~part.fm*, *tpl~preface.fm*

Topic Template (topic_1.1.template.fm)

Added fmx-masterpage marker definition.

Bug Fixes / Minor Updates

Updated "missing image" feature

The solution (added in 1.1.14) to eliminate the XML parser log messages about missing images has been updated. Instead of swapping all images for a default PNG, the temporary image replacement is done with an image of the same type. This eliminates the "missing filter" message that resulted from the previous implementation. A new "missing-images" folder now contains multiple "inline" and "break" images. The location of this folder defaults to a missing-images folder in the "Program Files" DITA-FMx folder, but can be set to another location by setting the GeneralImport/MissingImageDir parameter in the *ditafmx.ini* file. For details, see MissingImageDir in the INI-Only Settings topic.

Rotated table cells now remain rotated in generated files

If the "Conditionalize Data Elements" option is enabled, rotated cells were not remaining rotated in generated files. The cells now remain rotated regardless of this setting.

Language-specific book-build INI file now works properly

If the GeneralImport/UseLanguageTemplate parameter is set to "1", the book-build process will check for language-specific versions of the component templates. Additionally it will check for an use a language-specific version of the book-build INI file. For example, if the map's @xml:lang attribute value is set to "ja-jp", DITA-FMx will look for the file *ditafmx-bookbuild.ja-jp.ini*.

fm-xref can now use outputclass for format name

Added the GeneralExport/FmXrefUseOutputclass parameter to *ditafmx.ini*. If set to 1, specifies that fm-xref elements use the @outputclass attribute rather than the @type attribute to store the cross-ref format name. (Should be considered experimental. Currently defaults to “0” if this parameter is missing or set to a null string, but will default to “1” in DITA-FMx 2.0.)

Linebreak now properly maintained in lines element

The 1.1.14 update caused linebreaks in the lines element to include a line-break PI, but not the linebreak itself. This has been fixed.

Set Attributes functionality more sensible

When selecting an element after setting the attributes for another, the currently selected attribute remains selected if that attribute exists on the newly selected element.

Undo history now preserved

Clicking in the document window no longer breaks the undo history.

Xref and link references without a topicId now link to the root element

Xref and link elements whose href attributes include only a target filename will now properly reference the root element in that file.

Whitespace is properly stripped from elements within a section

Previously, with the Normalize Whitespace option enabled, whitespace within a section element was not stripped. It is now.

Whitespace is properly normalized for inline elements

Previously, for inline elements that terminated at the end of a line in the XML the whitespace that followed would be lost. It is now properly normalized.

Clipboard contents no longer lost

When creating a new file or opening an existing file, the clipboard contents remain available to paste after the operation has completed.

Apply Ditaval as Conditions “crasher” on Win7 appears fixed

Some Windows 7 systems would crash when the Apply Ditaval as Conditions command was run. This appears to be resolved now (fingers crossed).

Build Map from Outline “crasher” on Win7 appears fixed

Some Windows 7 systems would crash when the Build Map form Outline command was run. This appears to be resolved now (fingers crossed).

A PI that wraps to another line will crash FM during link processing

Strange but true. Has been fixed.

1.1.14 - 31 January 2012

New Features

Default structure applications are automatically installed

On a new installation of DITA-FMx, the default structure applications are installed on the first use of FrameMaker.

New CallClient APIs have been added

The “InstallApps” API is now available for use by other applications to automatically install structure applications. This API does not require the use of the FMx-Auto addon.

The “AssignIdToElem” API is now available for use by other applications to assign an ID to the specified or selected element in a document. This API does require the use of the FMx-Auto addon.

Integration with the FMx-Auto addon

This DITA-FMx update is the minimum version that can be used with the FMx-Auto addon. FMx-Auto is a separate plugin that enables the API in DITA-FMx to allow automated PDF (and other) publishing options.

Integration with DITA2Go

If you have DITA2Go installed, a “DITA2Go Project Manager” menu item will be available on the DITA-FMx menu. (This checks for the %OMSYSHOME% environment variable and the existence of the *d2gpm.exe* file at %omsyshome%\common\bin.)

Integration with SuiteHelp DITA-OT plugin

The *ditafmx-ant.xml* file has been updated to include a “suitehelp” target. If you have the SuiteHelp DITA-OT plugin installed (from Suite Solutions), you’ll be able to generate the SuiteHelp webhelp output from the DITA-FMx Generate Output dialog.

If you are upgrading DITA-FMx, in order to take advantage of this integration, in addition to installing the SuiteHelp plugin you’ll need to do the following:

- Copy the updated *ditafmx-ant.xml* file into your DITA-OT folder then run the DITA-OT *startcmd.bat* file and integrate the new *ditafmx-ant.xml* file by running: `ant-f integrator.xml`
- Update your *ditafmx.ini* file by adding a new “suitehelp” entry to the BuildFile section (be sure to update the “Count” parameter as well)

Structure Application Updates

None.

Bug Fixes / Minor Updates

Updated the version of the import/export client to 1.1.14

Previously, the two plugin DLLs were not in sync; they were updated separately as needed when their code was changed. In an effort to reduce confusion, both DLLs will now have the same version number regardless of any changes to the underlying code. The import/export client would have been version 1.1.12 with this release, but it is now set to 1.1.14 to match the authoring client DLL.

XML Read Report messages no longer report missing images in book build

Previously, images were reported as missing even though they would typically be relinked to the proper file location during the book-build process. Missing images are now replaced with a default “missing” image file on file open. This process prevents the XML Read Report window from displaying when no real problem exists. The default missing image is defined by the value of the image/@placement attribute, and is one of the two files (installed in the *FrameMaker\DITA-FMx* folder) *missing-image-break.png* or *missing-image-inline.png*.

Use ditavals in Book Build process without registering the ditaval file

The DitavalName parameter now accepts a file name in addition to the registered ditaval name. This simplifies the use of alternate ditaval files for book builds.

Added new Book Build setting for controlling conditions

New General/AttrAsCondDefaultState INI parameter defines the default Hide/Show state for all “fmx-” conditions. Significantly updated the related documentation on this topic, see Passing Map-level Metadata to the FM Book and Adding Map to Book Metadata Mappings.

New maps are opened in document view

For FM9/10 if the “Open maps in document view” option is enabled, newly created maps are now opened in document view.

Fixed problem with writing of duplicate document dictionary PIs

If the structure application template contained document dictionary entries, each time you open and save a file, the entries would get duplicated as processing instructions in the DITA file. Now if a file contains duplicate document dictionary entries, only one instance is written as a PI.

Added support for proper whitespace handling around conrefs and xrefs

Recompiled the FM10 import/export client with the updated FM10 FDK to incorporate fixes provided by Adobe regarding whitespace handling in FM10.

Corrected problems when using nesting separators in see-also index entries

Resolved data corruption when an indexterm used two or more “nesting separators.” Previously, if a see-also entry made use of more than two reference terms, indexterm elements would not import properly and would thus be lost on file save..

A semicolon can now be defined as a see-also nesting separator. When specifying this type of separator be sure to use a slash-semicolon (“\;”).

Conrefs to tables and footers now properly round trip with the XDocs CMS

Previously, conrefs to table and sub-table elements as well as footer elements were not being properly converted into CMS-based paths. XDocs refused to allow these files to be checked in.

Attribute selection in the Set Attributes dialog remains correct when switching elements

Previously, when using the Set Attributes command, changing the element selection would cause the selected attribute to change incorrectly. This no longer happens.

External links with a desc element format properly

Previously, if an external link contained a desc element, the text of the desc element was merged with the linktext element. This no longer happens.

Fixed problems with fragref and synnoteref elements

When fragref and synnoteref elements are used, they no longer are assigned the @type, @scope, and @format attributes (invalid for these elements). Also, when synnoteref is written to XML, it is now correctly written as an empty element. Note that there are still some EDD and functional issues with the syntaxdiagram child elements, however they should all round trip properly. (These remaining issues will be addressed in a future release.)

Because fragref and synnoteref are specializations of the xref element, the @type, @scope, and @format attributes would be added when written to XML, to prevent this from happening, a new *ditafmx.ini* setting has been added to the INIOnly section. Use the SuppressXrefAttrs parameter to specify elements that should not have these attributes added. If this parameter is not set, this value defaults to "fragref synnoteref".

syntaxdiagram/title is not treated as a figure title

Because syntaxdiagram inherits from fig, its title was moved when the MoveFigureTitles option is enabled. A new ditafmx.ini setting has been added to prevent this from happening. INIOnly/IgnoreFigTitleProcessing specifies a space-delimited list of fig-based elements to ignore. By default this is set to "syntaxdiagram".

Linebreak PIs are no longer added to preformatted elements

By default, a linebreak processing instruction is added to all linebreaks entered into the document. These are not needed and in fact redundant when included in elements based on the pre element (such as screen and codeblock). These PIs are no longer added to pre-based elements by default.

There are two new *ditafmx.ini* parameters provided to control the line-break PI feature. In the GeneralExport section, the WriteLineBreakPIs parameter can be set to 1 or 0 to enable or disable the writing of linebreak PIs (if not set this defaults to 1, enabled). Also in the GeneralExport section, the WriteLineBreakPIsInPre parameter can be set to 1 or 0 to enable or disable the writing of linebreak PIs within pre-based elements (if not set this defaults to 0, disabled). These parameters must be manually set in the *ditafmx.ini* file.

1.1.13 - 6 June 2011

New Features

None.

Structure Application Updates

Book XSLT (bookmap2fmbook.xsl)

- Updated $href translation to replace backslashes with an underscore to properly support maps created with default FM DITA (lines 707 and 996).
- Added topicgroup to the list of elements that were unwrapped (line 1068).

Bug Fixes / Minor Updates

Eliminated Japanese error messages in FM10 book build processing.

Replaced the resource file in the FM10 import/export client with the proper English version, so now all messages during the book-build process are in English rather than Japanese.

The “Move fig/title” book-build feature now moves the fig/desc.

If a figure includes a desc element, that will also be moved to follow the image if the “Move fig/title” option is enabled for book-build processing.

The Hide Conditionalized Content book-build feature now works properly.

Previously this option didn’t appear to be working; now it does.

Properly handles xrefs with @type=’fm:’.

FrameMaker 9 and 10 create xrefs that contain a type attribute of “fm:”. Previously this was assumed to be a format name and resulted in an unresolved cross reference. Now if the type attribute contains this value, it is ignored when opened with DITA-FMx.

The PageStartSide book-build INI parameter allows chapters with uneven pages.

When the PageStartSide parameter is used in a section within the book-build INI file, it will now properly generate chapters with uneven pages.

Tables no longer overlap with the content that follows.

Added a new bookbuild INI setting BookBuildOverrides/ReimportTemplate, if set to 1 enables the "Reimport Template" feature which, reimports the template from the “current” file on each component in the book. This should be used with caution since when enabled in previous releases was causing Frame to crash. So far this seems to be working fine and fixes the table overlap (as well as other formatting) issues.

Test for invalid marker elements on save.

When saving a file, test for invalid marker elements to ensure that the markers have the proper EDD element definition (should be a marker object rather than a container). If a marker element is defined as a container, the save will be prevented (if the save is allowed to happen, FrameMaker will crash). This is a common error when converting from unstructured FM files to structure.

1.1.12 - 29 March 2011

New Features

Support for FrameMaker 10.

DITA-FMx can now be installed on FM10. Remember that DITA-FMx 1.1 supports the DITA 1.1 specification, so even though native FM10 supports DITA 1.2, you cannot use DITA 1.2 features with this version of DITA-FMx. DITA-FMx 2.0 is under development and will support DITA 1.2.

Support for round-tripping of line breaks as PIs.

If the GeneralExport/WriteLineBreakPIs INI parameter is set to 1 (the default), line breaks (SHIFT+ENTER) entered into running text will be stored as processing instructions (PIs) in the XML. When opened in FrameMaker (with DITA-FMx), these PIs will convert into line breaks. For details, see the “WriteLineBreakPIs” description in INI-Only Settings.

Document dictionary items are stored in the DITA file as PIs.

When spell-checking a document, the “Allow in Document” button saves the current word in the document dictionary; this is typically lost when working with XML files. DITA-FMx stores the document dictionary as processing instructions so that these words will be available for future spell checks on that document.

Added a user interface for defining doctype/application mapping.

The Options dialog now includes an Edit button that lets you define a doctype/application mapping for those who want to define separate structure applications for each topic type. This is especially important for FM10 support.

Added new auto-prolog and new file name building blocks.

Four new building blocks have been added: <$FMX_USERNAME>, <$FMX_FULLNAME>, <$OS_USERNAME>, and <$OS_COMPUTER-NAME>. These building blocks are useful when the “RegInfo” values in the *maker.ini* file are not tied to the actual user.

Added support for accessing Help from a URL.

The DitaFMxGuide and DitaReference INI parameters can now specify a URL to access online Help from the web rather than a locally installed Help file. Added the DitaRefTargetPath parameter. For details, see the “DitaReference” description in INI-Only Settings.

Added the Open DITA-FMx Folder and Open Console Log commands.

Two new commands were added near the bottom DITA-FMx menu, **Open DITA-FMx Folder** and **Open Console Log**.The **Open DITA-FMx Folder** command opens a Windows Explorer window on the DITA-FMx folder to allow easy access to files. With Windows Vista or 7, this folder is under the user’s “app data” area. **Open Console Log** provides quick access to the *consfile.txt* file which contains the text of the messages printed to the console window.

Structure Application Updates

Map template (map_1.1.template.fm)

- Booklists items (in frontmatter and backmatter elements) no longer uses the WingDings font for the element labels.

Map EDD (map_1.1.edd.mif/fm)

- Changed the general rule for booklists elements (toc, indexlist, figurelist, etc.) so the fm-reflabel element is not required.

Book import XSLT (bookmap2fmbook.xsl and expandOrig.xsl)

- The *bookmap2fmbook.xsl* file was updated and *expandOrig.xsl* file was added to address some map to book conversion errors.

Book component template (gentpl~indexlist.fm)

- Deleted an errant tab from the heading. This was causing bad formatting in the generated TOC

Bug Fixes / Minor Updates

Index see/see-also forced sorting works for all levels of index entries.

The automatic addition of forced sort strings to see/see-also entries works for all levels of those entries. You should not add your own forced-sort strings. For more information, see Working with Indexterms.

Revised the book-build processing order to ensure consistent results.

Under certain conditions (with specific options enabled or disabled), some of the book-build features were undone or improperly built, due to the order of feature processing. We have revised this order to ensure consistent and correct results. Specific features that were known to be affected are: fmx-variable creation (via the ImportAttrsAsVars INI parameter), variable rebuilding (the “Rebuild Variables” book-build option), and table title relocation and creation (the “Move table/title” book-build option). Book-build start and end time is now printed to the console window. Other features may also have been affected.

The UseOutputclassForType bookbuild INI parameter now controls pagination.

If the General/UseOutputclassForType parameter in the *ditafmx-book-build.ini* file is enabled, top-level map elements that use the @outputclass attribute to specify an alternate template, can now make use of a like-named section to define the pagination and numbering for the resulting component.

Related links to dita-rooted files are now created during a book-build.

The root topic within dita-rooted files can now be the target of a related-link defined by relationship tables.

Saved view settings are applied to new topics.

If the Restore Saved View Settings option is enabled, those settings will now be applied to newly created topics.

INIOnly/LastReferencedElement parameter moved to the “temp” INI.

The LastReferencedElement parameter is not a user setting, and changes each time the Reference Manager is used. This value, previously stored in the *ditafmx.ini* file, is now stored in the *~fmx-temp.ini* file.

Updated handling of default values in INI files.

Some INI parameters, in particular ArchiveCommandLine, were not properly set to valid default values when those parameters were set to a null value. All parameters have been reviewed and when a null value is not valid, it is now set to the appropriate default value.

Authorization code corrections.

When entering the authorization data, user names can now contain non-alphanumeric character. Also, you no longer need to be running FrameMaker “As Administrator” to enter the authorization code.

1.1.11 - 10 January 2011

New Features

Enables “proper” handling of booklists elements.

The INIOnly/UseBooklistPlaceholder INI parameter enables the “proper” handling of frontmatter and backmatter automatic list generation where the @href attribute is empty rather than requiring placeholder files. For details, see the “UseBooklistPlaceholder” description in INI-Only Settings.

Added support for language-specific book-build INI files.

The INIOnly/UseLanguageTemplate INI parameter enables support for alternate language book-build INI files based on the value of the topic/@xml:lang attribute. For details, see the “UseLanguageTemplate” description in INI-Only Settings.

Added support for table cell rotation.

Rotated table cells will round-trip properly and are rendered properly in generated book components. For details, see Working with Tables.

Enhancements to the book-build process.

Five new parameters have been added to the book-build INI file. In the General section, the AttrAsCondPrefix parameter lets the user specify an alternate prefix for conditions generated from map attributes. In the BookBuildOverrides section the MoveFigId and MoveTableId parameters support proper referencing of table titles and figure titles, the BreakToInline parameter allows for better indenting of images, and the NormalizeConditions parameter eliminates the inconsistent show/hide setting and inconsistent condition indicator messages when updating a book. For details, see Book-Build INI file.

New DITA File dialog pre-selects last used element template.

When an element template is used to create a specific topic type, that same template will be pre-selected when that topic type is created again.

Added “Merge Para Tags” command.

The Merge Para Tags command ensures that all paragraph tag names in all currently open documents, exist in all of those documents. For more information, see Merge Para Tags.

Structure Application Updates

Book Structure Application

- Added a Chapter component template (tpl~chapter.fm), just a copy of the Book template, but a nice convenience.

Book EDD (book_1.1.edd.mif/fm)

- Added fm-ditabook/@xml:lang
- Changed fig/@id type to UniqueID
- Added fm-figuretitle to fig general rule
- Added fm-figuretitle element definition
- Changed table/@id type to a UniqueID
- Changed fm-tabletitle/@id type to UniqueID
- Added context rules to title element definition to apply appendix and part title formats eliminating the need for separate appendix and part component template EDDs.
- Added second set of context rules to fig element definition to support indenting of images.

Book template (book_1.1.template.fm) and component templates

- Changed autonum in note.danger para format to “danger” (was “fastpath”).
- Added figure.title.bottom and figure.title.wide.bottom para styles (for fm-figuretitle element).
- Removed “Fixed” line spacing from figure.anchor and figure.anchor.wide para styles to support the revised handling of indented images.
- Import updated Book EDD

Book DTD (fmxbook.dtd)

- Added @xml:lang to fm-ditabook

Book XSLT (bookmap2fmbook.xsl)

- Added "translate" to strip CRs from attribute values assigned to fm-ditabook element (CRs are introduced when map metadata wraps in the XML file)
- Added @xml:lang attribute to fm-ditabook
- Strip colons from FM filename (fmfile variable) in 2 places
- Added new template to process booklists/* map elements that have no @href value (no associated file)
- Added new template to handle data and data-about elements in topics separately from maps (they get stripped from a map [book] but not from topics)
- Added support for @print='no' (exclude from output) in 2 places

Map template (map_1.1.template.fm)

- Changed font family to Wingdings on topicref.lead.char character style

Topic EDD (topic_1.1.edd.mif/fm)

- Changed fig/@id type to UniqueID
- Changed table/@id type to a UniqueID

Topic template (topic_1.1.template.fm)

- Import updated EDD

Bug Fixes

Significant improvements to handling of graphic overlay objects.

The time required to open and save files that contain graphic overlay objects has been reduced significantly.

data and data-about elements are not deleted during book-build.

The XSLT import script no longer deletes the data and data-about elements from topics during a book-build.

topicref/@print='no' is now honored in book-build.

Topic references in a map that have the @print attribute set to "no" are now omitted from the generated book files.

fm-xref references to "moved" table and figure titles work properly.

Cross-references created with the fm-xref element will resolve properly even when the "Move title" options are selected in the book-build options.

All figure titles are rendered properly in the generated FM files.

The first figure title is no longer deleted from each chapter when the "move" option is used.

fm-link insertion no longer produces an error message.

Inserting an fm-link in FM9 no longer generates an invalid error message.

RelLinkType parameter is no longer ignored in the book-build INI file.

When specifying the related link type in the book-build INI file, the RelLinkType value will is now honored.

The Rebuild Variables in book-builds feature works properly now.

Selecting the Rebuild Variables option in the Book Build Options dialog or using the RebuildVars parameter in a book-build INI file will now enable previously "prepared" variables to be rebuilt in the generated FM files.

1.1.10 - 25 October 2010

New Features

Added Save View Settings command.

The "Save View Settings" feature introduced in version 1.1.09 operated on file save, and wasn't very easy to use. This update adds this feature as a new command so you can easily specify when to save the view settings.

Restore Previous View Settings option renamed to Restore Saved View Settings.

Renamed the option to align with the new like-named command. If this setting is enabled, the view settings saved with the Save View Settings command are applied to files as they are opened.

Save View Settings includes attribute display options.

The Save View Settings command now saves the attribute display option setting.

Structure Application Updates

None.

Bug Fixes

Open Maps in Document View (FM9) works for submaps.

Opening a submap from a topicref in a map will open and honor the Open Maps in Document View setting.

1.1.09 - 4 October 2010

New Features

Better integration with Windows Vista and Windows 7.

DITA-FMx does not need to be run "As Administrator."

Added support for SDL Trisoft CMS.

This version of DITA-FMx is designed to integrate with the SDL Trisoft CMS.

Added the option to restore the previous view settings.

On file open, restores the document zoom value and the visibility of borders, text symbols, rulers, grid lines, and element boundaries. See "Restore Previous View Settings" in the DITA Options topic.

Import fm-ditabook attributes as variables and condition states.

Attributes on the fm-ditabook element can define values that are passed to the book components as FrameMaker variables. These attributes can also be used to control the show/hide state of conditions in the generated book components. See Adding Map to Book Metadata Mappings.

Added support for pre and post book-build scripts.

The book-build INI file can now specify pre and/or post build script file names to assist in automating setup and cleanup tasks. These file names specify batch or executable script files, and are not used to run an FDK client or FrameScript. See Book-Build INI file.

Override default book-build settings via the book-build INI file.

The book-build INI file can now override the default values set in the Book Build Options dialog. See Book-Build INI file.

Allow specification of component templates via the @outputclass attribute.

You can now set the outputclass attribute on topicref or topicref-based elements in maps, and that attribute value will be added to the fm-ditafile element in generated files in a book. This value can be used instead of the "mapelemtype" value in the book-build INI file to specify templates and properties of book components. See Using the outputclass attribute as a "mapelemtype" value.

Added the "substring extraction" option to New File building blocks.

When using new file name building blocks, in addition to being able to extract the first *N* characters of the string you can now specify a range of characters to extract. See "New File Name Format" in the New File Options topic.

Added the option to open DITA maps in document view (FM9 only).

See "Open Maps in Document View" in the DITA Options topic.

Provide a user-definable template file name delimiter.

Allows modification of the file name delimiter used in new file templates, component templates, and generated list templates. See "TplDelimChar" in the INI-Only Settings topic.

Structure Application Updates

Topic and Book EDDs

Corrected a typo in the comments: "index_3" has been changed to "indent_3".

Topic and Book EDDs and templates

Added the following sl list paragraph styles:

- sl.ul.entry.bullet
- sl.ol.entry.bullet
- sl.entry.bullet
- sl.step.bullet
- sl.ul.bullet
- sl.ol.bullet
- sl.bullet

Topic and Book templates

glossdef, removed Keep w/ Next

glossterm, removed Keep w/ Next, add Keep w/ Previous.

glossterm.runhead, removed Keep w/ Next.

Component templates

Added ix-see and ix-seealso character styles to gentpl~toc.fm file.

Added support for appendix and part TOC entries.

Added new tpl~appendix.fm template.

Book EDD and DTD (fmxbook.dtd)

Added numerous new attributes to the fm-ditabook element.

Added @outputclass attribute to the fm-ditafile element.

Book EDD

Added fm-indexterm as child of title element.

Added support for title.4 as a 6th level heading.

Book template

Added <*$chapnum*>- to prefix figure title and table title paragraph tags.

Removed hyphenation from title.0.

Added ix-see and ix-seealso character styles.

Book XSLT

Overall updates to improve handling of various folder structures.

Added example code to add fm-ditabook attributes from map and bookmap metadata.

Map EDD

Added @id attribute to the relcolspec element.

Bug Fixes

The number of ditaval files supported by the Ditaval Manager has been increased.

The number of ditaval files that can be listed in the Ditaval Manager is restricted to the length of the aggregated string of all of the ditaval names (just the file name, not the extension or path). This length was 256 and is now 1024. The new length should accommodate 50 ditaval files with an average length of 20 characters.

Create Archive command now available from Resource Manager.

The Create Archive command can now be run on a map open in the FM9 Resource Manager.

Flatten conref fixes.

Nested conrefs will now flatten properly, and selected individual conrefs in an XML file will now remain flattened after the XML file is reopened.

Nested conrefs to the same file resolve properly.

Conrefs that reference an element within the same file will now properly resolve nested conrefs within the referenced element.

The "fmdpi" feature is only used for raster images.

The "fmdpi" value is no longer set for vector graphics. Because you can't set the DPI of a vector graphic, this caused sizing problems.

Xrefs that contain Unicode characters are no longer deleted on file open.

Entering alternate xref text that contains Unicode characters, will now properly round trip through FrameMaker.

The Open All Topicrefs command opens topic references that include IDs.

If the @href in a topicref-based element includes the topic ID, the referenced file will now open correctly.

The Open All Files in Book command opens DITA map files.

If a workbook contains references to a DITA map, that map file will now be opened as expected along with other topic files.

Added support for structure applications with a read-only template.

Read-only template files will no longer cause problems. The same goes for a *structapps.fm* file that is read-only.

FrameMaker variables support additional characters.

FM variables in topic files can now contain alphanumeric, plus dot, dash, and underscore characters (other characters will cause errors at file open).

Ampersands can now be used in xref/@href attributes.

When an ampersand is used in an @href it will be converted to "&amp;" on file save.

Set Attributes dialog displays sorted attribute names.

The attribute names in the Set Attributes dialog are now sorted.

Properly supports xrefs that contain high-ascii characters.

An xref with alternate link text containing non-English characters (such as an "ä") will now properly resolve and render when the document is reopened.

The @class value is deleted when an element type is changed.

When changing an element's type (using the "Change" button in the Element Catalog) the @class value will often get hard-coded to the previous element's value. This causes problems for DITA-OT processing. Now, when you change an element, the @class value is deleted (unless you explictly turn this off with the INIOnly/StripClassAttribute setting in the *ditafmx.ini* file).

1.1.08 (1.1 release) - 20 October 2009

New Features

New commands

Added the **Reference Report** command. Generates a report of all resolved or unresolved references in the current map or file as well as any references in referenced files.

Added the **Create Archive** command to generate a ZIP archive of the current file and all referenced files.

Added the **Flatten Conrefs** command to unlock conrefs in FM files.

Added the **Reset Status** command to remove the status attribute values.

Updated support for indexterm elements

The index-see, index-see-also, and index-sort-as elements properly convert to FM marker syntax on import and export back to DITA elements on file save. A subdialog has been added to the Options dialog that allows you to customize the formatting on the imported marker syntax.

The indexterm elements now import as fm-indexterm elements. If you have complex indexterm elements that contain child elements other than indexterm, index-see, index-see-also, and index-sort-as, you can disable the indexterm to fm-indexterm conversion (in Options) in order to preserve the native DITA structure.

Support for graphic overlay objects in anchored frames

Textual callouts and graphic objects within an anchored frame will now round-trip from FrameMaker to DITA and back. The data to define these objects is stored in the DITA data element.

Reference Manager dialog allows browsing for files on disk

The Reference Manager dialog now allows you to browse files on disk in addition to those currently open in FrameMaker. You can add multiple folders to the “File Location” list to locate elements to reference (for xrefs, links, and conrefs).

Also, instead of showing all possible elements the Reference Manager dialog only shows those elements that have id attributes in the target file.

Support for multiple marker types

The fm-data-marker element is a special element that is available only within FrameMaker; it is converted into a regular data element on save to DITA. When inserting an fm-data-marker element it inserts as a marker of any type that you specify. On save to DITA, the datatype attribute is set to “fm:marker,” the name attribute is set to the marker name, and the value attribute is set to the marker text. When you reopen this DITA file, data elements with a datatype of “fm:marker” will convert into markers of type fm-data-marker. A data element whose datatype attribute is not “fm:marker” will import into FrameMaker as a standard container element type.

Support for optional columns in properties tables

According to the DITA specification, a properties table can have one, two, or three columns. In order to support a varying number of columns in a simpletable-based table, the Element Mapping dialog was added to the Options dialog.

Added support for bookmap elements

The map authoring interface provides support for the topic reference elements (topicref, chapter, appendix, etc.). When inserting a topicref-based element that contains an href attribute, you are prompted to specify a target filename.

Enhanced map to book processing

The **Generate Book from Map** command handles conversion of a bookmap into an FM book in a similar way to the conversion of a map. All “top-level” topic references are converted into an FM file, and any child topicrefs are appended to that file. Any topic references within a frontmatter or backmatter wrapper element are considered “top-level” topic references and become FM files. Additionally, if your bookmap uses a part element to organize chapters, the file referenced by the part will become a separate FM file, and each chapter will be considered a top-level topic reference and will be added to the book after the generated part file.

Provide the ability to include FM binary files in the generated book. This allows you to create a book that includes both DITA files as well as unstructured FM files. Particularly useful for including a well designed title page but could be useful for other purposes as well.

Numerous options are available that affect processes run on the generated book. This includes adding related-links from reltables, and moving the fig/title to the end of the fig element (among numerous other options).

Faster reference resolving

Instead of needing to open each referenced DITA file in FrameMaker to resolve any nested references, this reference resolving is done "on disk" before the file is opened, and only the appropriate files are opened and resolved as needed. This results in much faster file opening times for files that make heavy use of references. For example, one small file that previously took 11 seconds to open, now takes 4 seconds.

Better Whitespace Normalization

This update now handles whitespace cleanup better than before. If you are working on a multi-editor environment, you should no longer see "<WHITESPACE>" nodes or extra spaces where they aren't needed.

Updated Set Attributes command

The Set Attributes command now allows specification of predefined values for attributes of the "String" type. If values are specified in the filtering-groups.ini file for a "String" attribute, those values will display in a scrolling list.

XDocs 2.0 integration

Developed the XDocsFMx connector to provide a seamless document management, authoring, and publishing environment.

New options

Added **Use doctype/application mapping** option to support the use of topic-based DTDs (and structure applicaitons) instead of the combined model provided by the ditabase DTD. (You'll need to create your own structure applications to support the topic-based DTDs, DITA-FMx only provides the ditabase-based app.)

You can now specify that IDs are generated as a GUID (globally unique id) or a QUID (quasi unique id, the previous shorter and most always unique value).

Added **Auto smart-quotes** similar in function to the **Auto smart-spaces** option.

Added **Use fmdpi for New Images** option. Automatically sets the "fmdpi" value to the DPI selected when inserting an image.

Added the **Conditionalize data and data-about** option. This allows you to hide or flag data and data-about elements if needed.

Added **indexterm to fm-indexterm** option to enable/disable the round-tripping of DITA-based indexterm elements into FM-based markers and syntax. (This option is enabled by default.)

DITA-OT environment setup parameter is now set via the DITA Options: External Apps dialog.

Added handling for simpletable tables with multiple cell element names (specifically properties table). See Element Mapping dialog in DITA Options

Added an Index Options dialog for control of the import and export of index-see and index-see-also elements.

Added the Book Builds dialog to define the automated processes that are run on a newly generated FM book file.

Moved **Wrap in dita element** option to the New DITA File dialog.

Now automatically sets the status attribute on new and changed elements with the **Set @status for new/changed elements** option.

Added the **Use wide Reference Manager** option. This provides a wider Reference Manager for those that need it.

Structure Application Updates

Updated to support DITA 1.1

Added DITA 1.1 elements to all structure applications.

Default Topic application is less “book-like”

Because DITA-sourced content may be used for many types of output, we have removed many of the print-specific layout and formatting features from the default Topic application. This helps authors to focus on the content rather than the formatting, and results in topics that are more useful in all deliverable formats. The main typeface used is now Verdana.

Added FM-specific elements

Added fm-indexterm as a marker element that has the same attributes as the standard indexterm element. The indexterm element is now a “container” element. (For backwards compatibility, if a structure application contains an indexterm element that is defined as a “marker” element type, it will be used instead of the fm-indexterm element.)

Added the fm-data-marker element as a “marker” element type.

Changed the fm-topicreflabel element to fm-reflabel since it is now used by elements other than topicref. (For backwards compatibility, if a structure application contains the fm-topicreflabel, that will be used in place of the fm-reflabel.)

Customization and localization (topic and book applications)

To lessen the level of effort in EDD/template customization and localization, some elements that formerly received prefix text directly from the

EDD now have that text applied via a paragraph tag or set of paragraph tags in the template. With a plan to include more elements in the future, this version took this approach with the note and permissions elements.

The note prefix text is now applied using a set of template paragraph tags. With the exception of "other," each note type attribute value now maps to a similarly named note paragraph tag. For example, a note type of "danger" maps to a paragraph tag named note.danger.

To further help support both localization and customization, the first and left indents for note are set in the EDD using variables, indent_1, indent_2, index_3, and indent_4. Changing those variable values will change the indents for both the note and the screen elements.

The permissions element is now mapped to a set of prolog.permissions paragraph tags.

The prefix text for the example element has been dropped.

The codeblock element has a set of paragraph tags for its various contexts. This set of tags all begin with name codeblock (codeblock.ol.ul.first, codeblock.ol.ul, etc.).

The syntaxdiagram element is now mapped to a syntaxdiagram paragraph tag.

The prolog element is now mapped to a prolog paragraph tag.

Color definitions have been altered to be more consistent between the topic and book applications.

Map/Bookmap application

Various levels of indentation have been added for the topicref-type elements that occur in bookmaps.

The relcolspec element now displays attribute values for type, collection-type, and linking when in reltables, and collection-type and linking in other contexts.

The indexterm element is now mapped to an index character tag.

The vrm element's prefix text has been modified.

Bug Fixes

Baseline offset now supported

The baseline offset property of an image cannot be set through the read/write rules file. To set this property you must use the BaselineOffset parameter in the INIOnly section of the *ditafmx.ini* file.

image/alt element now supported

The image/alt element now round-trips from DITA to FM. While in FrameMaker, the content of the alt element is stored in the image/@alt attribute. To edit the “alt” text, edit the @alt attribute value. Note that any child elements of the alt element or attributes on the alt element are discarded when the file is opened in FM. A warning message is written to the console window on file open if attributes or child elements are detected.

UNC path improvements

Working with files in folders on a UNC mapped network drive (“\\server-name”) will now result in the relative paths being saved properly (as relative paths) rather than as absolute UNC paths.

xref/link issues

All xref and link @href values are now written with backslashes.

New file naming issue

If the file name for a new file contains a period, it was incorrectly assumed to represent the file extension. Now if a file does not end with “.dita” or “.xml” the default file extension is appended.

Long application names display properly in Options dialog

A structure application name that wraps to multiple lines in the structure application definitions file now show the entire name in the Options dialog.

Changing the DPI of an image that uses the fmdpi feature will update properly

Images that use the fmdpi feature now properly update the fmdpi value when the DPI of an image is changed.

topicref/@navtitle is not added if @navtitle has been removed from the EDD

When building topicrefs, if the EDD has been customized to remove the navtitle attribute, that attribute is longer automatically added (causing an invalid structure).

Where Used now searches in maps

The Where Used command now properly reports instances of topics in maps.

Search in Files now ignores URLs

The Search in Files command now ignores URLs when scanning for files to search.

1.00.28 - 16 December 2008

Bug Fixes

Apply Ditaval as Conditions command fixes.

Properly applies conditions to empty elements, specifically markers and image elements.

The Clear All Existing Conditions option now works in a more expected manner.

Console message revisions.

Removed console message when generating a book from a map that reported images not in the project scope.

1.00.27 - 28 November 2008

Bug Fixes

Apply Ditaval as Conditions command applies overlapping conditions.

Now this “really” works properly.

Conref updates.

Conrefs to elements within a table cell no longer have the extra “end of para” mark that visually adds an extra empty line to the cell.

Conref paths that reference a higher level directory (“../”) are built properly now (in cases where directory names were very similar, the paths were not properly built).

FM cross-refs.

All fm-xrefs and fm-links that specify an undefined cross-reference format will now render with a label (previously the first one of each format type was blank).

Map from outline.

Attempting to generate a map from an outline file that is unsaved will no longer crash FM.

1.00.26 - 31 August 2008

New Features

Search in Files command matches on partial attribute values.

The Search in Files command now allows for matching on partial attribute values. This is especially useful for locating elements that have multiple values applied to their filtering attributes.

Bug Fixes

Apply Ditaval as Conditions command applies overlapping conditions.

The Apply Ditaval as Conditions command now applies overlapping conditions as required by attribute settings.

Generate Book from Map no longer saves XML files as FM.

In rare circumstances the Generate Book from Map command would save the XML file as a binary FM file making it impossible to continue. This no longer happens.

WMF images are now properly handled.

When adding a WMF image, it was not properly shrinkwrapped and the href value was not set until file save. WMF files are now handled the same as other image types

Windows Vista problems fixed.

On Windows Vista, inserting an xref no longer causes FM to crash.

Element templates must be closed to be used.

If you try to use an element template for a new file when the element template is open, you now receive a warning that you must close the template before it can be used.

Build Map from Outline handles invalid para styles.

If a paragraph style in the outline document references a style that doesn’t map to a topic type FM will no longer crash.

Conref problems resolved.

A conref to an xref will no longer cause FM to hang. Conrefs can now be made to simpletable elements (and related table types).

1.00.25 (1.0 release) - 7 July 2008

New Features

FrameMaker 8 support.

Now supports FrameMaker 8 and all of its Unicode functionality.

Full support for link elements.

Link elements in the related-link section of a topic are now managed the same way xref elements have been.

Search in Files command.

Provides the ability to search for content in files within a folder (and sub-folders) or in files referenced by a DITA map. The search criteria can be a mix of textual content, element or attribute names, or attribute value.

Where Used command.

Generates a report listing all files that reference the selected element or current topic.

Set Attributes command.

Provides quick and easy access to setting attributes on elements. In particular, this command makes use of the FrameMaker “Strings” attribute type and allows you to select one or more default values that are applied to the attribute.

Updated New DITA File command.

The New DITA File dialog now lets you enter the topic or map title and automatically generate a proposed file name based on your specification (in the New File Options dialog). You can also optionally select an element template to insert predefined structure and content to the new file. If needed, you can specify a folder name along with the file name and that folder will be created if needed.

Integrated ditaval support and management.

The Ditaval Manager provides an easy to use interface for creating and managing ditaval files. These files can be used to apply conditional filtering to FrameMaker books and documents as well as passed to the Open Toolkit for filtering of the generated content.

Handles “pretty-printed” XML files.

A new option strips padding (spaces and tabs) from files on import.

Auto-Prolog feature.

A new option lets you specify certain prolog data to automatically add or update on file creation and file save.

Build Map from Outline command.

Creates a DITA map and optionally DITA topic files from a simple FrameMaker document.

Build 'WorkBook' from Map command.

Generates FrameMaker book file that contains all of the DITA files referenced by a DITA map and any sub-maps. This command facilitates the use of FrameMaker's commands that iterate over files in a book (such as spell checking and search).

Open All XML Files in Book command.

Intended to be used with the "WorkBook," this command opens all of the XML files in a book and provides the option to resolve references or not in the opened files.

Topicref labels in DITA maps.

A new option allows you to choose the type of content displayed in the topicref label. You can see the title, the file name or both.

New DITA File updates.

The New DITA File command is now available from the File menu in addition to the DITA-FMx menu. When creating a new DITA file, you can now overwrite an existing file, and create new folders.

ID attribute validation.

When manually changing an ID attribute value, a warning is displayed if it is invalid.

Added Xref to Hyperlink command.

This command converts DITA-based xrefs and links into FrameMaker Hyperlinks that are live hyperlinks in generated PDF files.

Variables persist through the Map to Book process.

Two new commands have been added, Prepare Variables and Rebuild Variables, which make it possible for FrameMaker variables that are used in DITA files to be available as live variables in the generated book files.

Reference Manager "remembers" last selected element type.

When you use the Reference Manager, it now defaults to selecting the last referenced element type.

Added option to disable coloring of conrefs.

If the coloring of conrefs causes problems for your output tools (WebWorks in particular doesn’t like it), you can now disable this in the Topic or Book applications.

Now recognizes the “include” ditaval action.

The Ditaval Manager and the Apply Ditaval as Conditions command now recognize the “include” ditaval action value.

Added Check for Updates command.

The Check for Updates command was added to the menu as well as a setting in the Options dialog to specify the frequency to automatically check for updates.

Added “break conref” functionality.

If you need to convert a conref into editable text (and sever the connection to the source element), delete the conref attribute value and double-click the conref.

Added special image-handling features.

The “fmdpi” feature maintains alternate image sizes within FrameMaker. Add the value, “fmdpi:<DPI>” to the image/@otherprops attribute (where <DPI> is the DPI value). See the Working with Images topic for more info.

The default alignment for new images is now defined by the default value of the image/@align attribute.

Added API calls.

FMxVer, FixBookRefs, and LoadReferences.

Added the fm-* elements to the DITA Reference.

Now when you use Alt+F1 to get help on the selected element, it will work for the “fm-*” elements as well.

Structure Application Updates

Changed “String” to “Strings” type for filtering attributes.

To allow easy use of the new filtering groups feature of the Set Attributes command, the attribute type for platform, product, audience, and otherprops has been changed to “Strings” in the Topic and Map applications.

Added a user-settable method of round tripping graphics as non-cropped.

A “don't crop” read/write rule has been added to the topic and book rules files. By default it is commented out, but by enabling it, graphics can round-trip without FrameMaker resetting them to “Cropped” during import.

Added formatting support for more elements within a fig.

Some formatting support for lists (ul and ol), p, dl, and note has been added and now includes formatting for contexts where fig has its attribute expanse set to page. Also, a list whose compact attribute is set to “yes” within a fig will now format as a horizontal list.

Line below Task is more reliably drawn.

The line indicating the completion of a Task now occurs in more contexts.

Corrected format of Topic label text in reltable heading.

The “Topic,” “Reference,” “Concept,” and “Task” text that automatically appears in the column heading of a reltable is now properly left-aligned in its cell.

Topicmeta formats for author and keyword are now more consistent.

Author element text now properly aligns with the other elements in topicmeta. And, keyword elements receive more consistent formatting where multiple keywords occur.

Book template DITA-Comment and DITA-Prolog conditions display default is now properly set.

With the DITA option set to conditionalize Prolog or Comment elements, when a DITA file is opened that contains comment or prolog elements, the template default is now to Hide those elements. (For Topic and Map templates, the default behavior is still to Show those conditions.)

Bug Fixes

Processing Instructions after the document close tag.

Opening an XML file that has a PI (processing instruction) after the document's closing tag no longer crashes FM when you save that file.

Proper handling of colons in the forced sort area of an index entry.

Colons in the forced sort area of an index entry (within square brackets) are no longer treated as level separators when saving to XML. The forced sort content is written to the last indexterm element.

fm-xref and fm-link element fixes.

Fixed problem where fm-xref and fm-link elements did not properly reference sub-topic elements.

New file command properly creates topics with specialized title elements.

If your specialized topic type uses a “title” element with a name other than “title” it properly uses that specialized element.

Clipboard content is not lost when opening a new topicref.

If you copy something to the clipboard, then open a topic by clicking a topicref, that content is still available for pasting.

Conrefs that specify no file name now resolve to the current file.

When the conref attribute specifies no file name (as in `conref="#topicid/elemid"`) the conref looks for "topicid/elemid" in the current file.

Topicrefs that reference a topicid now process correctly.

Updated the book processing XSLT file so it can handle topicrefs that reference a file and a topicid (topicref/@href='filename.xml#topicid').

On file open, a missing image now triggers the "missing image" dialog.

When opening a file, if an image can't be found, the default "Select image file" dialog is now displayed.

Replacing an existing image properly updates the href attribute.

If an existing image is replaced through the FrameMaker interface, when the file is reopened the new file name is still properly specified.

Column widths in simpletable and choicetable elements.

These elements now round trip properly.

Tables in generated files are full width.

When building a book, tables now fill the width of the text column.

Moved INI parameters around a bit.

The following INI parameters have been moved from one section to another (as indicated):

General -> INIOnly/AutoLoadTopicrefs

General -> INIOnly/XrefElements

General -> INIOnly/StructappsFile

General -> INIOnly/FmXrefElem

General -> INIOnly/FileOpenClient

General -> INIOnly/DitaHelpKeys

General -> INIOnly/DitaReference

General -> DitavalDefaults/DitavalName

General -> DitavalDefaults/DitavalExcludeConditionName

General -> DitavalDefaults/DitavalExcludeConditionNameType

General -> DitavalDefaults/DitavalExcludeConditionVisibility

General -> DitavalDefaults/DitavalFlagConditionName

General -> DitavalDefaults/DitavalFlagConditionNameType

General -> DitavalDefaults/DitavalFlagConditionVisibility

GeneralImport -> General/CheckForComments

Other misc. fixes.

Various cleanup and bug fixes.

0.02 - 18 December 2007

New Features

Book-building component is now included.

The ability to generate a FrameMaker book from a DITA map is now included with DITA-FMx.

Added ditaval filtering for authoring and output.

The **Apply Ditaval as Conditions** command lets you apply ditaval filtering to the document being authored, and the **Use Selected Ditaval File** option in the Generate Output dialog lets you specify a ditaval file for use with a DITA-OT target.

New "Auto Smart-Spaces" option makes it easier to work with code in documents.

If enabled, this option automatically disables and enables the FrameMaker "Smart Spaces" option when the insertion point is moved into and out of a preformatted text element (one based on the "topic/pre" class).

Child elements and line breaks can coexist in preformatted elements.

The new option, "Fix line breaks in topic/pre elements," properly allows for child elements within preformatted ("topic/pre") elements. If this option is enabled, line breaks are no longer lost when a child element is before or after a line break. (This overcomes the native FrameMaker bug/feature.)

The character sequence "]]>" can be used in FrameMaker.

Typically, you can't include the characters "]]>" within a FrameMaker document saved to XML. DITA-FMx now overcomes this standard limitation.

Specify the number of reference levels to resolve.

When you open a DITA file (and the auto-loading reference options are enabled), DITA-FMx resolves all conrefs and xrefs in all referenced files.

This new option allows you to specify the number of reference levels to resolve (typically 2 is plenty). For documentation sets that make extensive use of references, this can significantly speed up the time it takes to open topic files.

Pressing ESC while files are resolving will interrupt the process.

If you need to abort the file resolving process, press ESC.

The Reference Manager dialog now defaults to the last referenced file.

Instead of defaulting to the current file, the Reference Manager now defaults to the most recently referenced file.

External Xref dialog now provides External or Peer options for the scope attribute.

The scope attribute of external xrefs were previously assigned the value of "external," this value can now be set to either "external" or "peer."

When xrefs are created they are now populated with the proper type, format, and scope attributes.

Xref elements created and modified through the Reference Manager are now assigned the proper type, format, and scope attributes.

Set up environment variables before running the Ant script.

The EnvironmentSetup INI-only parameter lets you specify a batch file to run before the Ant script in order to properly set up the environment for an OT build.

Updated support for topicmeta element in a map.

The EDD and template have been updated to provide a nicer rendering of topicmeta data. Also, a new option was added to the Options dialog to allow conditionalizing of topicmeta on file open.

TOC and Index templates provided for easy book-building.

Two template files are now provided with the Book application making it easy to set up a complete book.

Structure Application Updates

Topic Application.

<note> - Formatting changes to correct some indention issues.

<ol>, <ul>, <sl>, <li>, and <sli> - Formatting changes to correct indention and "red text" issues.

<xref> - Character format added for context scope= "external" (link.external).

<apiname> - Now formats as a text range.

Book Application.

Same updates as those for the Topic application.

<fm-ditabook> - Added <fm-subditamap>.

<title> - Modified formatting of <title> to provide support for maps of maps. Modified syntax- and group-related context labels so that they'd not be picked up in TOC generation that went to Level4.

Map Application.

Corrected copyright and legal info.

<topicmeta> - Added formatting for all <topicmeta> elements when option is invoked. Added DITA-Topicmeta condition to template.

Bug Fixes

Conrefs resolve properly.

All known problems with conrefs have been fixed, including conrefs to tables and conrefs to single block elements.

Xrefs that wrap over a line are no longer truncated on save.

Xrefs at the end of a paragraph now save properly.

Indexterms in book builds aren't duplicated.

Nested indexterms in a book build resulted in the duplication of all but the top level entry, this no longer happens.

Indexterm elements are no longer saved with the hard-coded class value.

This made it impossible to specialize the indexterm element.

DITA-Comment and DITA-Prolog conditions are no longer saved as PIs.

Because conditions are saved as Processing Instructions, the DITA-Comment and DITA-Prolog conditions were saved as well. This caused a problem when the “Conditionalize element on file open” options were used, causing the condition to be set even when the options were disabled.

In the DITA Options dialog, if an application is not selected, it shows as empty.

Previously, this would default to the first application in the list.

Table format and tgroup/@outputclass value are now in sync.

Changing one updates the other.

The clipboard contents are now saved before creating a new DITA file.

Previously, the clipboard contents were lost when the New DITA File command was used.

The Open All Topicrefs command properly processes all topicref files.

Now all topicrefs are processed.

No longer displays warning about comments when resolving references.

This warning now only happens on file open (if the option is enabled).

Elements with general rule of "<TEXT>" now conref properly.

Any elements with the lone general rule of "<TEXT>" were saved incorrectly when included by conref. A temporary fix was to change the general rule to "(<TEXT>)*" but this fix is no longer needed.

The Reference Manager dialog now properly highlights the selected topic.

Previously when double-clicking an existing xref, the current file was not selected.

Long application names now display properly in the Options dialog.

Previously, if an application name was so long that it wrapped in the *structapps.fm* file the name would be truncated in the Options dialog.

0.01 - 20 August 2007

Initial Beta release.

Index