oManual plug-in User's Guide For DITA Open Toolkit

Contents

Preface

Preface

About the DITA to oManual Converter

The org.omanual.dita2omanual plugin enables the conversion of applicable DITA task content into the IEEE-approved oManual XML format for import into iFixit.com for publication.

Best suited for "repair manual" type content with rich media *http://www.ifixit.com/*.

DITA to oManual plug-in features

This plug-in has following features.

stylesheets	This plug-in contains DITA to oManual stylesheets written to XSLT 2.0 conventions.
Supports DITA tasks	Converts maps to category document; tasks to guide entries and main guide documents.
feature	Description

Limitations

Currently converts only DITA tasks (since procedures are the primary feature of oManual).

Content conversion aims to cover oManual's allowed HTML subset. DITA content that does not align with this subset is currently mapped into a div that identifies it as unsupported content (which mainly means just that a better idea is needed on how to port the content over more gracefully!).

An Ant build task has been provided, and it works in principle for the tasks themselves--each is converted into its respective guide. However, the map itself does not generate the procedure shell. It's mainly a matter of Ant setup and testing, and is open for any contributor to help complete the integration. Once enabled, builds may be set up that apply conditional processing as part of the transform. As currently effected from the command line, you would have to have done that pre-processing in a separate step using the eclipsecontent transtype as suggested elsewhere.

List of Figures

List of Tables

Chapter

1

Plug-in Installation

This section describes the plug-in installation for the DITA Open Toolkit.

The plug-in zip file

The plug-in zip file contains the following files and folders.

```
+---com.omanual.dita2omanual
|     |
|     +--- dita (sample DITA source content for testing)
|     |      |
|     |      +--- other
|     |      |
|     |      +--- ptc
|     |      |
|     |      +--- taskbook
|     |            |
|     |            +--- DITA instances and image files
|     |
|     +--- doc
|     |
|     +--- omanual (sample oManual source content for testing)
|     |      |
|     |      +--- xsd
|     |      |
|     |      +--- examples
|     |
|     +--- xsl
|     |      |
|     |      +--- dita2omanual.xsl (and child stylesheet files)
|     |
|     +--- build.xml, integrator.xml, plugin.xml, revision.txt
|
+--- plugin.xml, integrator.xml, build_dita2omanual.xml
```

To install the plug-in into DITA Open Toolkit, follow the next instructions.

Downloading DITA Open Toolkit

You can download DITA Open Toolkit from the following URL: *https://sourceforge.net/projects/dita-ot/files/*. There are many kind of archives. The "full_easy_install" version is handy because it contains all needed jar files.

After downloading archive file, unzip it into the appropriate folder. From now this folder is called simply [DITA-OT] for convenience.

This plug-in works with DITA-OT 1.6.3 or later.

Installing dita2omanual Plug-in

Follow the next instructions.

1. Copy org.omanual.dita2omanual folder to [DITA-OT]\plugins folder.
2. *Copy index-data folder to [DITA-OT]\samples folder.
3. *Copy all of the batch file in the root folder to [DITA-OT] folder.
4. Make [DITA-OT]\out folder.
5. (Assuming Windows!) From the file explorer, click [DITA-OT]\startcmd.bat file. The command window titled "DITA-OT" opens.
6. From command window enter the following command. This command integrates PDF5 plug-in into DITA Open Toolkit.

   ```
   ant -f integrator.xml
   ```

   Figure 1: Integrating new plug-ins into DITA Open Toolkit
7. Close the command window.

Testing the plug-in

This plug-in contains test data. Follow the next instructions.

1. If the DITA-OT command window is already open, skip the next step.
2. (Assuming Windows!) From the file explorer, click [DITA-OT]\startcmd.bat file. The command window titled "DITA-OT" opens.
3. This converter may be run as either a java command line application or as an Ant build script.
4. You can test run_en.bat batch files from this window. This batch file has no parameter.

   sample command

   Figure 2: Example of running a batch file in DITA Open Toolkit

   The target oManual files and log file will be generated in the [DITA-OT]\out folder.
5. Discuss how to stage this output for the next step, importing into iFixit.

Plug-in And Stylesheet Parameters

This plug-in has plug-in general and plug-in specific command-line parameters.

General Plug-in Parameters

Following table describes DITA-OT general plug-in parameters.

Property name in build file **Name in the stylesheet**	**Default value**	**Notes**
args.input		Specify input map file path.
dita.temp.dir		Specify the folder path where DITA-OT will create temporary files generated during the transformation process.
clean.temp	yes	Specifies whether to clean up temporary folder.
dita.input.valfile		Specifies .ditaval file for filtering.
output.dir		Specifies the folder path where DITA-OT outputs the results.

Property name in build file **Name in the stylesheet**	**Default value**	**Notes**
transtype		Specify "pdf5" to invoke this plug-in.

Figure 3: General plug-in parameters

Plug-in Specific Parameters

This plug-in and stylesheet has the following parameters. The plug-in parameter are defined in the org.omanual.dita2omanual\build.xml as property. The stylesheet parameters are defined as xsl:param in org.omanual.dita2omanual\xsl\dita2fo_param.xsl.

Property name in build file **Name in the stylesheet**	**Default value**	**Notes**
output.pdf	[map file name].pdf	Specifies output PDF file name with extension.
xsl.file	org.omanual.dita2omanual\	
xsl\dita2fo_shell.xsl		Specifies main stylesheet file path.
style.def.file	PRM_STYLE_DEF_FI LE	
..\config	\default_style.xml	Specifies default style definition file.
alt.style.def.file	PRM_ALT_STYLE_D EF_FILE	
..\config\[language- code]_style.xml		Specifies the alternate style definition file path. By default the file name is automatically made from xml:lang attribute value of the map top element.
no		This parameter is only used with I18n Index Library when making zh-CN manual.
yes		Specifies whether to make link for "See" and "See Also" entry in the index page. If this parameter is 'yes', the author must make corresponding indexterm for "See" and "See Also".
xml.lang	PRM_LANG	The xml:lang attribute value of the map top element. Specifies different xml:lang value for the map.
output.draft.comment	PRM_OUTPUT_DRAF T_COMMENT	
output.required.cleanup	PRM_OUTPUT_REQU IRED_CLEANUP	
no		Specifies whether to output <draftcomment> element content.
no		Specifies whether to output <required-cleanup> element content.

Property name in build file **Name in the stylesheet**	**Default value**	**Notes**
yes		Specifies whether to format <dl> as block. If this parameter is 'no', <dl> is formatted using table.[1]
yes		Specifies whether to honor toc='no' attribute or not. [2]
yes		If this parameter is "yes", plug-in outputs the start message using <xsl:message>.
user.input.dir.url	PRM_MAP_DIR_URL	URL of the input map folder. Specify input map (bookmap) folder URL.

Figure 4: Plug-in specific parameters

[1] Traditionally <dl> is formatted as block. However DITA introduced title element <dlhead>. If you use dl/dlhead element, it is better to format it using table.

[2] In the DITA Version 1.1 Architectural Specification p.25, the toc attribute is described to control navigation output. This will be applied for HTML or other output. However current other DITA to XSL-FO stylesheet honors this attribute. This parameter controls that stylesheet should honor toc attribute or not.

Chapter 2

Command-line Format And Parameters

After starting command window with [DITA-OT]\startcmd.bat, you can enter the following commands to invoke this plug-in.

Note that this plugin currently runs as a file-to-file transformation process, so it is not sensitive to preprocessing for a particular set of ditaval conditions, for example. For content that needs to be filtered prior to conversion into oManual format, we recommend using the eclipsecontent transtype. This transtype produces normalized DITA files with all indicated conditionality applied, ready for use as the input to that particular new set of oManual output.

Java command-line formats

Without reference to an XML catalog for DITA:

```
C:\DITA-OT1.8.M2>java -jar lib\saxon\saxon9.jar -s:plugins/
org.omanual.dita2omanual/dita/taskbook-omanual.ditamap -xsl:plugins/
org.omanual.dita2omanual/xsl/dita2omanual.xsl > _temp/test.xml
```

Properly referencing the DITA Open Toolkit's provided XML catalog for DITA:

```
java -Dxml.catalog.files=catalog-dita.xml -Dxml.catalog.verbosity=1
 net.sf.saxon.Transform -r org.apache.xml.resolver.tools.CatalogResolver
 -x org.apache.xml.resolver.tools.ResolvingXMLReader -y
 org.apache.xml.resolver.tools.ResolvingXMLReader -s:plugins/
org.omanual.dita2omanual/dita/taskbook-omanual.ditamap -xsl:plugins/
org.omanual.dita2omanual/xsl/dita2omanual.xsl > _temp/test.xml
```

Note: Saxon-based transforms from the command line can make use of the DITA Open Toolkit's embedded XML catalogs for validation. However, the catalog is not implicitly available even in the enhanced "startcmd" shell environment, hence the need for the added command line parameters. For future work, these parameters can be added to a run-time Ant target, which at that time will be documented in the next section.

See the Sourceforge reference, *XML Catalogs*, for more background on using XML Catalogs in the Saxon command line environment.

Ant command-line formats

Pattern:

```
ant -l [logfile path] -Dtranstype=dita2omanual -D[property]=[value] ...
```

Run the sample build task:

```
ant -f plugins\org.omanual.dita2omanual\run-sample.xml
```

Summary of provided Ant tasks in this plugin:

`ant -f plugins\org.omanual.dita2omanual\docs_pdf.xml`	Generate this documentation as a PDF using the default FOP processor.
`ant -f plugins\org.omanual.dita2omanual\sample_omanual.xml`	Generate oManual output from the Toolkit's default Taskbook sample (no oManual enhancements).

`ant -f plugins\org.omanual.dita2omanual\run-sample.xml`	Generate oManual output from the provided Taskbook sample in this plugin (enhanced to exercise some oManual features).

Chapter 3

Plug-in process flow

This plug-in produces oManual output from DITA input as follows. (The rest of this process has not been updated yet.)

```
                    DITA-OT
```

Figure 5: Plug-in process flow

definition list to organize discussion about the defined process points

Chapter 4

Stylesheet Structure

The DITA to oManual stylesheets have following names and contents. The stylesheet files are located in the \xsl folder.

Table 1: Stylesheet file and contents

File Name	Contents
dita2omanual.xsl	Calls the map and topic processors.
map2category.xsl	Walks the map, converts structure to the oManual category format.
topic2guide.xsl	Per task, converts the steps into oManual guide steps.
extras.xsl	Support functions.

Chapter 5

Customizing the Transforms

Any DITA topic type may be processed by these transforms, but only tasks produce sensible results as oManual guides. As this project evolves, we expect contributions that will improve on the content mapping from both arbitrary and specialized DITA resources.

This plugin is based on standard DITA-aware match patterns, therefore it can be extended by conventional overrides for specialized DITA content.

References to existing override how-to guides:

- From DITA-OT docs...
- From bloggers...
- Books: Leigh and Kimber citations

Chapter
6

Stylesheet Messages

{short description}

The DITA to oManual stylesheets outputs the following messages. The suffix character of the message number means:

The suffix is "I"	This is an information message.
The suffix is "W"	This is a warning message.
The suffix is "F"	This is a fatal error message. XSLT processor will stop processing after this message has outputted.

Table 2: Stylesheet messages

Message	Contents	Discussion
These are informational and often include diagnostic queries as params.		
010	HTML-formatted info about the category.	discussion
011	Unmatched element needing consideration.	origin passes name of unmatched element
000	Skipping topic	origin passes name of skipped element
001	Skipping concept	origin passes name of skipped element
002	Skipping reference	origin passes name of skipped element
003	Skipping a grouping topicref	origin passes name of skipped element
004	Outputting	origin passes output filename info
These are validation conditions that need to be met, either in source prep or post migration cleanup.		
005	Banner image. A local image; use empty string if not specified.	discussion
006	Author. Please identify an author (and corresponding @id).	discussion
007	Time required. Please fill in a suggested time value.	discussion
008	Difficulty. Please fill in suggested level of difficulty, skill, or experience.	discussion
009	Category.	Note: map to prolog/metadata/category for now (note that DITA allows multiples)

Appendix A

Principles of Operation

Notes for Developers about the conversion tool's architecture and algorithms. As you learn things that other developers and contributors should know, please log those observations here.

{paragraph content}

General

The goal for this conversion is getting as much information from the DITA document into the oManual format (input for the Dozuki publishing system) as easily as possible. Therefore this transform uses pull templates to model the full schema. When using pull-based transforms, some elements in the result tree could end up empty, but this has the advantage of faithfully enforcing the required structure of the result.

Comments

1. The image field of a category represents the banner image. Dozuki import will do the size generation as part of the import process, so authoring best practice is to cite the largest image size available.
2. Tables and definition lists in DITA currently have no equivalent markup in oManual for conversion to. Currently, a definition list will be turned into a series of paragraphs with the terms in bold markup. Tables are skipped at present for need of a better idea on how to handle their conversion into existing oManual semantics.
3. new

Appendix B

DITA to oManual edge case conversions

This topic documents the implicit business rules for the various defaults/messages/workarounds for missing or unmatched data. These mappings come from several sources:

- The data required by the schemas.
- Allowed values and patterns for normalizing typed data like dates or model numbers.
- Other data as observed or contributed.

Required contexts and default dispositions (category)

XPath:	Disposition:
category/@locale	Default to 'en' and update from topic/@xml:lang if present.
flag/@title	Default to 'In Migration' and update from map/@status (not a direct DITA equivalent, so generate message)
guide/@id	Retrieved from DITA map/topic id, or hashed as needed.
guide/@subject	Currently mapped from the topic's title, which is not exactly equivalent. DITA source may have other associational conventions.
guide/@type	Default to 'installation'. Long term, update from keyword match in the title, etc. (such as "Installing the Frammitz")

Required contexts and default dispositions (guide)

XPath:	Disposition:
guide/@locale	Default to 'en' and update from topic/@xml:lang if present.
guide/image	Required banner image. Typically a local image. If an image doesn't exist, set it to an empty string. **Use i.e. outputclass="banner" on the image element to designate it as a banner image.** **Note:** This value is not normally available or obvious in DITA content; can pass non-null default by param (guide_default_image_filename) to override empty defaults.
guide/@type	One of 'teardown | replacement | repair' (observed; use the list obtained from finalized spec schemas.) If a type role hasn't been defined in markup, set it to an empty string. **Use i.e. outputclass="repair" on the task element to designate its type.** **Note:** This value is not normally available or obvious in DITA content; can pass non-null default by param (guide_default_type) to override empty defaults.

XPath:	**Disposition:**
media/@type	One of 'video | image | imbed' according to the contained resource.
embed/@type	One of 'photo | video | link | rich' (oEmbed standard values)
embed/@version	'1.0' (oEmbed sole value)
tool/@name	Tool name (good candidate for retrieving from a controlled vocabulary)
part/@name	Part name (same rationale for validated values)
part/@quantity	Part quantity (1 or more; if empty on import, will be set to 1)
flag/@title	Synonym for 'status', and suggests a need to be validated against an external enumerated values list.
document/@url	An atomic property. Any object (link, xref, download, image) with an address URL should require that property. **Note:** If the document URL is local, it should be a relative path to the attached document.
prerequisite/@id	Identity for referencing **Note:** This can be anything, this will be replaced on upload.
prerequisite/@locale	Same rule as guide/@locale. In effect, this is a typed link to a guide typed by what points to it. (blown head)
prerequisite/@path	Address for referencing **Note:** Relative path to the prerequisite guide xml file
step/@number	A pregenerated sequence counter starting from 1 (string type)
video/encoding/@url	Address of video resource (like a param of DITA's object element). In fact, all encoding attributes are required

Messages for user action

Msgnum:	**Explanation:**
{msgnum}	{desc}

Appendix C

Helping the Community

General discussion about project participation.

- gitHub participation
- oManual IEEE standards participation
- iFixit community development for DITA contributors
- Other?

Appendix D

Writing DITA Tasks for oManual

Background

- Differences between DITA tasks and oManual guides
- General DITA best practices (for example, *Writing Effective Task Topics in DIT"A*
- Best practices for writing DITA repair procedures
    - oManual content is strongly media oriented. DITA procedure writers should follow oManual guidelines for how to represent engaging repair content.
    - DITA content writers should not use source markup that an oManual content author would not use.