[vertical list of authors]

[cover art/text goes here]

Contents

DITA Open Toolkit

Navigation title: DITA Toolkit Introduction

The DITA Open Toolkit is a reference implementation of the OASIS DITA Technical Committee's specification for DITA DTDs and Schemas. The Toolkit transforms DITA content (maps and topics) into deliverable formats, including: XHTML, Eclipse Help, HTML Help, and JavaHelp.

DITA release notes

Introduction

Release 1.1.2 is a maintenance release to fix defects and make patches based on release 1.1.1.

But there are certain limitations and unfixed bugs in this release, such as,

- Bug 1343963 Blank index.html generated for ditamap contains only reltabe
- Bug 1344486 java.io.EOFException thrown out when reading ditaval file

Please check the current 'open' bugs on the SourceForge bugs tracker.

Changes

1. SF Bug 1297355: Multilevel HTML Help popup shows filenames
2. SF Bug 1297657: Update for Supported Parameters page
3. SF Bug 1304859: Toolkit disallows repetition of topic ID within map
4. SF Bug 1306361: Fatal error in published ditamap example
5. SF Bug 1306363: common.css not compiled with htmlhelp
6. SF Bug 1311788: DTD references not resolved
7. SF Bug 1314081: Fix catalog entries in catalag-ant.xml for OASIS DTDs
8. SF Bug 1323435: wrong system id for html output used in validation
9. SF Bug 1323486: HTML Help subterm indexes not sorted
10. SF Bug 1325290: JavaHelp output does not work for Russian
11. SF Bug 1325277: File missing from the map causes abend
12. SF Patch 1253783: dita2fo-links relative hrefs
13. SF Patch 1324387: In xslfo, groupchoice var prints extra | delimiter
14. SF RFE 1324990: Installation Guide

Note

SourceForge bugs, patches, and RFEs can be found in SourceForge bugs, patches, and RFE tracker.

- Bugs tracker: http://sourceforge.net/tracker/?group_id=132728&atid=725074
- Patches tracker: http://sourceforge.net/tracker/?group_id=132728&atid=725076
- RFE tracker: http://sourceforge.net/tracker/?group_id=132728&atid=725077

DITA release history

This document lists major changes and new features by release.

DITA OT release 1.1.2

Release 1.1.2 is a maintenance release to fix defects and make patches based on release 1.1.1.

But there are certain limitations and unfixed bugs in this release, such as,

- Bug 1343963 Blank index.html generated for ditamap contains only reltabe
- Bug 1344486 java.io.EOFException thrown out when reading ditaval file

Please check the current 'open' bugs on the SourceForge bugs tracker.

Changes

1. SF Bug 1297355: Multilevel HTML Help popup shows filenames
2. SF Bug 1297657: Update for Supported Parameters page
3. SF Bug 1304859: Toolkit disallows repetition of topic ID within map
4. SF Bug 1306361: Fatal error in published ditamap example
5. SF Bug 1306363: common.css not compiled with htmlhelp
6. SF Bug 1311788: DTD references not resolved
7. SF Bug 1314081: Fix catalog entries in catalag-ant.xml for OASIS DTDs
8. SF Bug 1323435: wrong system id for html output used in validation
9. SF Bug 1323486: HTML Help subterm indexes not sorted
10. SF Bug 1325290: JavaHelp output does not work for Russian
11. SF Bug 1325277: File missing from the map causes abend
12. SF Patch 1253783: dita2fo-links relative hrefs
13. SF Patch 1324387: In xslfo, groupchoice var prints extra | delimiter
14. SF RFE 1324990: Installation Guide

DITA OT release 1.1.1

Release 1.1.1 is a maintenance release to fix defects and make patches based on release 1.1.

For patch 1284023, we are changing the name of the jar lib file from dost1.0.jar back to dost.jar because we believe we need to keep the jar file name consistent through various releases.

Changes

1. SF Bug 1196409: HTMLHelp output does not reference CSS
2. SF Bug 1272687: extra "../" link generated by topicgroup
3. SF Bug 1273751: revision flag using unavailable pictures
4. SF Bug 1273816: Index generation doesn't cope with multilevel well
5. SF Bug 1281900: Unnecessary comment and href typo
6. SF Bug 1283600: unecessary space in document cause invalid parameter of Ant
7. SF Bug 1283644: multipul document($FILTERFILE,/) doesn't work (XALAN)
8. SF Patch 1251609: pretargets xsl directory needs to use ${dita.script.dir}
9. SF Patch 1252441: Files in temp directory not deleted before build
10. SF Patch 1253785: Inline images in dita2fo-elems
11. SF Patch 1284023: change the name of jar file and remove the version name

DITA OT release 1.1

Release 1.1 is a major release to add new functions, fulfill new requirements, make some function enhancements and fix bugs over release 1.0.2.

1. **Adaptation to the new OASIS DITA standard**

   Release 1.1 implements the new OASIS DITA 1.0 standard for DITA DTDs and Schemas.

   DTDs of the previous release locate in the directory **dtd/dita132** and schemas of the previous release locate in the directory **schema/dita132**.

2. **Transformation to troff**

   Release 1.1 supports new troff output. Troff output looks like Linux man page output.

3. **XML catalog support**

   An XML catalog, which can consist of several catalog entry files, is a logical structure that describes mapping information between public IDs and URLs of DTD files. A catalog entry file is an XML file that includes a group of catalog entries. If you want to know more about XML catalog, please refer XML Catalog .

   A catalog entry can be used to locate a unified resource identifier (URI) reference

for a certain resource such as a DTD file. An external entity's public identifier is used for mapping to the URI reference. The URI of any system identifier can be ignored.

4. **Topicref referring to a nested topic**

   The href attribute of the topicref is extended to quote a nested topic in a dita file.

   For example, in previous releases, href attribute is set like: href = "xxx.dita"; in release 1.1, href attribute can be set like: href = "xxx.dita#abc.dita".

5. **Globalization support**

   Release 1.1 supports over 20 popular languages within the content of dita files. And it also provides translation function for DITA keywords to over 20 languages. Currently this globalization support fully applies to Eclipse Help and XHTML transformations, and partially applies to other transformations.

6. **Accessibility support**

   Accessibility support is now partially applies to PDF and XHTML transformations.

7. **Eclipse Content Provider Support**

   Please refer to Eclipse Content Provider for detail information.

8. **Index information in output**

   The output of HTML Help and Java Help transformations contain index information now.

9. **Mapref function**

   Mapref refers to a special usage of the `<topicref>` element as a reference to another ditamap file. This allows you to manage the overall ditamap file more easily. A large ditamap file can thus be broken down into several ditamap files, making it easier for the user to manage the overall logical structure. On the other hand, this mechanism also increases the reusability of those ditamap files. If you want to know more about mapref, please refer Mapref .

10. **TOC generation for Eclipse Help transformation**

    TOC generation now supported in transformation to Eclipse Help. Eclipse.

11. **Helpset generation for Java Help transformation**

    Helpset generation now supported in transformation to Java Help.

12. **New parameters supported in Java commands**

    In Java commands: /indexshow, /outext, /copycss, /xsl, /tempdir.

13. **New parameters supported in Ant scripts**

    In Ant scripts: args.indexshow, args.outext, args.copycss, args.xsl, dita.temp.dir

Other Changes

1. SF bug 1220569: Add XML Schema processing to DITA-OT
2. SF bug 1220644: Prompted ant--image does not link for single topic to PDF
3. SF bug 1229058: Add schema validation loading file for processing
4. SF RFE 1176855: Ant must be run from toolkit directory
5. SF RFE 1183482: Copy pre-existing html to output dir
6. SF RFE 1183490: Provide argument to specify the location of temp dir
7. SF RFE 1201242: override capability

DITA OT release 1.0.2

Release 1.0.2 is a maintenance release to fix defects and adds some minor

enhancements in release 1.0.1.

Changes

1. SF Bug 1181950: format attribute should be set to 'dita' for dita topic
2. SF RFE 1183487: Document the usage of footer property
3. SF RFE 1198847: command line interface support
4. SF RFE 1198850: architecture document update
5. SF RFE 1200410: need explanation for dita.list
6. SF RFE 1201175: XML catalog support
7. SF Patch 1176909: Add template for getting image URI

DITA OT release 1.0.1

Release 1.0.1 is a maintenance release to fix defects and adds some minor enhancements in release 1.0.

Changes

1. Committer: maplink.xsl doesn't generate related links for second level referred topic
2. Committer: avoid infinite loop of conref
3. SF Bug 1160964: Can't point above the directory which contains the map file
4. SF Bug 1163523: Broken XPath expression in mappull.xsl
5. SF Bug 1168974: useless DRAFT param in FO transformation
6. SF Bug 1173162: generate null internal link destination in fo transformation
7. SF Bug 1173164: Not correctly use document() in dita2fo-links.xsl
8. SF Bug 1173663: All base directories are DITA-OT 1.0
9. SF Patch 1163561: XLST match patterns test for element names
10. SF Patch 1165068: FO hyperlinks and FOP-generated PDF bookmarks
11. SF Patch 1174012: Modification to sequence.ditamap

DITA OT release 1.0

The initial release of the Open Sourced DITA Toolkit introduces major architectural changes from the previous, developerWorks version of the Toolkit.

New features

1. A new, Java-based processing architecture that supports single-threaded execution throughout.
2. Ant-based orchestration of the processing environment, from preprocessing to transformation to any required post-processing.
3. A pre-processor core that supports conditional processing and conref resolution.
4. Map-driven processing that generates links for transformed topics.
5. A new DITA to HTML transform that replaces the previous topic2html_Impl.xsl core transform. This new core is based on requirements for high-volume usage within IBM for the past several years.

Ant-driven processing means that you can integrate the DITA processing tools into a seamless pipeline within supportive environments such as Eclipse.

The DTDs and Schemas in this version are based on those in the previous dita132 package with bug fixes. The DITA OS Toolkit will later support the OASIS 1.0 specification in its public review form.

DITA history on developerWorks (pre-Open Source)

Versions of the toolkit prior to Open Source are in the developerWorks XML Zone at this address: DITA Downloads Change logs for those versions are within the Readme files in each distribution.

DITA futures

Activity on the DITA Open Toolkit project will revolve around maintenance (bug fixes),

enhancements (new function based on prioritized requests), demos and experimentation (sandbox activity), and community support (forums, etc.).

DITA Open Toolkit 1.0 is a major upgrade from its predecessor, the developerWorks version known as "dita132." Because this is a new project with a new name. a new home, and largely new code, and because it is considered production-level code for XHTML output, the project numbering has been initiated at 1.0 for the first built release. The 1.0 version of code is still based on the dita132 DTDs and Schemas.

Major improvements from dita132 include:

- A new processing architecture that includes a new preprocessing stage
- Full conref resolution in the preprocessor
- Full conditional resolution (filtering and flagging) in the preprocessor
- Second pass transformation into final output formats
- Use of Ant and Java for the processing sequence and utility code
- A high-quality transform for XHTML output based on code that IBM has tested and used for the past 5 years
- Translated libraries for generated text in 47 languages (accessed by region and country code)

Future plans:
Future development activity of the DITA Open Toolkit is based on the end goal of providing a complete reference implementation for all core output transforms. The anticipated order of work based on current prioritizations (post 1.0) will be:

- 1.1: Develop the currently demo-level FO transforms to support production-level, book-like functionality with a generic format that can be easily interfaced for particular corporate styles and branding. This will involve working with the OASIS DITA Technical Committee to validate and endorse the bookmap specialization of DITA map. (roughly matching the DITA TC 1.1 plan, based on the OASIS DITA 1.0-level DTDs and Schemas, expected to be a Spec in this timeframe). This version will be based on the OASIS DITA 1.0 level of DTDs and Schemas.
- 1.2 (roughly): Develop the remaining demo-level help tools to support production-level output for these output formats: Eclipse help with plugin support, HTML Help, and JavaHelp. Also other new help formats as prioritized for this release (such as manpage, QT Assistant, etc.). (roughly the 1.2 plan, based on OASIS DITA 1.0-level DTDs and Schemas with any fixes known at that point)
- 1.3 (roughly): Develop migrators for OASIS updates that might impact existing DITA source. Other requirements as identified, such as styling layers, custom package building from the project, interfaces to translation standards such as XLIFF, and so forth.

The project will use the SourceForge RFE tool to accept new requirements. These will be prioritized for placement into plan according to the process in the Development Process document.

Tested platforms and tools

Navigation title: Tested Platforms and tools

See which tools and platforms have been used in testing the DITA processing system.

The DITA processing system has been tested against the following platforms and tools:

Tested OS:	Windows, RedHat Linux 9
Tested XSLT processor:	Xalan-J 2.6, Saxon 6.5 **Note:** XSLT 2.0 standard is not supported yet, don't use XSLT 2.0 engines. For example: Saxon 8.x.

Tested JDK:	IBM 1.4.2, SUN 1.4.2
Tested Ant:	Ant 1.6.2

Using DITA transforms

The core transforms of the DITA Toolkit represent the “Reference Implementation” for processing the standard DITA specification as maintained by OASIS Open.

Pre-process

A pre-process is done before the main transformation. The input of Pre-process is dita files and the output of Pre-process is also dita files. But the output is in temp directory. Pre-process is the basic for the main transformation, it handles several different processing before the main transformation. Without pre-process, dita topics and map can still be transformed into different outputs, but the features in pre-process such as resolving conref attribute are not available.

Available core transforms

A core DITA transform is the basic set of templates that support all the elements of a topic. This set is the basis for the following processing of any specialized element. Core transforms handle one topic instance, or nested set of topics, at a time. The DITA Toolkit provides these core transforms:

dita2xhtml.xsl
DITA topic to HTML page transform.

dita2fo-shell.xsl
DITA topic to XSL Formatting Object page transform.

Available special output formats

Additional map-driven tools support transforming sets of topics into special output formats, including:

Web page (map2htmtoc.xsl)
This transform generates a set of web pages with an index page that is ready to place on a Web site.

map2htmlhelp (map2hhc.xsl map2hhp.xsl)
This transform generates hhc and hhp file for the compilation of Html Help.

map2javahelp (map2JavaHelpToc.xsl map2JavaHelpMap.xsl)
This transform generates table of content and jhm file for Java Help.

map2eclipsehelp (map2elipse.xsl)
This transform generates table of content for help contents in Eclipse.

map2printout
Calls topicmerge to consolidate a set of topics into a single entity that is transformed into Formatting Objects (FO), which can be compiled into PDF.

Invoke the complete transformation

The complete transformation including pre-process can be excuted by the ant script. There are some examples of simple ant script in directory /ant. The ant target for the transformation which can be called is listed at Running Ant

Building DITA output with Ant

Ant is an open tool that uses the DITA processes to make producing DITA output easier.

Introduction of Ant

DITA provides a set of XSLT scripts for producing different types of documentations such as: help output in Eclipse, Java Help and HTML Help, web HTML pages and PDF file.

To make it easier to call these scripts, the DITA distribution now provides an experimental Ant tool to automatically build the DITA documentations, demos, and samples.

Ant is a Java-based, open source tool provided by the Apache Foundation to declare a sequence of build actions. Meanwhile, Ant is well suited for development builds as well as document builds.

It is unnecessary for Ant to set up a build environment to run the DITA XSLT scripts. To run the DITA scripts directly, see the DITA Readme document.

Note: The following instructions and the associated *build.xml* and *ditatargets.xml* files are for the Java 1.4.2, Ant 1.6.2, FOP 0.20.5, and Saxon 6.5.3 releases. These instructions are likely to need some adjustment for other versions of these components and for specific environments.

Setting up Ant

Navigation title: Setting up Ant

This topic guides you how to set up Ant environment properly.

Assume that you have already installed the Java Development Kit (JDK) and the XSLT processor before setting up the Ant.

Set up the Ant

1. Download and extract the Ant package file (available on http://ant.apache.org/bindownload.cgi) into a directory of your choice.
2. Set up environment variable.

If you use Windows,	**follow these steps.** • **Set the JAVA_HOME. `set JAVA_HOME=<jdk_dir>`** • **Set the ANT_HOME. `set ANT_HOME=<ant_dir>`** • **Set the PATH. `set PATH=%PATH%;<ant_dir>\bin`**
If you use Linux,	**follow these steps.** • **Set JAVA_HOME `export JAVA_HOME=<jdk_dir>`** • **Set the ANT_HOME `export ANT_HOME=<ant_dir>`** • **Set the PATH (`export PATH=$PATH:<ant_dir>\bin`**

3. If you have installed optional output FOP to generate PDF output, see DITA installation for detail information of setting up.

Running Ant

Navigation title: Running Ant

After setting up the Ant environment, you can build the DITA output by running `ant` command.

Here are some samples to explain how to use Ant to build sample output in the DITA directory.
Note: To run the Ant demo properly, you should switch to the **DITA installation directory** under the command prompt.

- You can build all demos in the DITA directory.

  Input `ant all`

The building process will create an **/out/** directory and puts the output files in subdirectories that parallel the source directory.

- You can also rebuild specific part of output of the DITA sample files.

  You need to remove part of the output first by specifying a "`clean`" target, and then rebuild the output. For example: To rebuild FAQ demo, input

  ```
  ant clean.demo.faq

  ant demo.faq
  ```

  Note: To find out the complete list of targets you can clean and build, check the *name* attributes for the target elements within the *build.xml* file. Or, input `ant -projecthelp` for information.
- You can also build assigned input to output in a default and easy way.

  Input `ant`

  Ant will prompt you for the input and output, and you need to input the directories of input files and output with correctly upper or lower case.

You can reuse the targets provided by the *conductor.xml* file in builds for your own DITA content by coping the *build.xml, conductor.xml, pretargets.xml, ditatargets.xml* and *catalog-ant.xml* files into a new directory and edit the *build.xml* to specify your DITA files. Refer to Ant tasks and tweaks for more information of those functions.

Note: To troubleshoot problems in setting up Java, Ant, Saxon, or FOP, you will get better information from the communities for those components rather than the communities for the DITA. Of course, if you find issues relevant to the DITA XSLT scripts (or have ideas for improving them), you are encouraged to engage the DITA community.

Ant tasks and script

This topic lists detailed Ant tasks and script.

The build process including pre-process can be called by using Ant script. There are four major Ant script files:

conductor.xml, *pretargets.xml, ditatargets.xml* and *catalog-ant.xml*.

conductor.xml

The main Ant script file includes the other three ant scripts and provides main targets for every output style.

Table1. General Parameter Table

Parameter	Description	Required
args.input	The path and name of the input file. This argument should be in the same upper or lower case with the filename on file system. **Note:** This parameter must be provided if `dita.input` and `dita.input.dirname` not be provided.	No
dita.input	The name of the input file . **Note:** This parameter must be provided if `args.input` not be provided. And this parameter must be used together with the `dita.input.dirname` parameter. The result of this combination is equivalent to use only the `args.input` parameter. It is an alternative way to specify the path and name of the input file.	No
dita.input.dirnam	The input directory which contains the input file. **Note:** This parameter must be provided if `args.input` not be	No

Parameter	Description	Required
	provided. And this parameter must be used together with the `dita.input` parameter. The result of this combination is equivalent to use only the `args.input` parameter. It is an alternative way to specify the path and name of the input file.	
dita.temp.dir	The directory of the temporary files. The default is 'temp'.	No
output.dir	The path of the output directory.	Yes
dita.extname	The file extension name of the input topic files, for example, '.xml' or '.dita'. The default is '.xml'.	No
args.xsl	The xsl file to replace the default xsl file. It will replace dita2docbook.xsl in docbook transformation, dita2fo-shell.xsl in pdf transformation, dita2xhtml.xsl in xhtml/eclipsehelp transformation, and dita2html.xsl in javahelp/htmlhelp transformation.	No
dita.input.valfile	The name of the file containing *filter/flagging/revision* information.	No
args.draft	Default "hide draft & cleanup content" processing parameter (“no” = hide them); “no” and “yes” are valid values; non- “yes” is ignored.	No
args.artlbl	Default "output artwork filenames" processing parameter; “no” and “yes” are valid values; non- “yes” is ignored.	No
clean.temp	The parameter to specify whether to clean the temp directory before each build. Only "no" and "yes" are valid values. The default is yes.	No

Table1. General Parameter Table for Tasks(dita2xhtml,dita2htmlhelp,dita2javahelp,dita2eclipsehelp)

Parameter	Description	Required
args.indexshow	The parameter to specify whether each index entry should display within the body of the text itself. Only "no" and "yes" are valid values.	No
args.copycss	The parameter to specify whether copy user specified css files to the directory specified by `args.csspath`. Only "no" and "yes" are valid values.	No
args.outext	The output file extension name for generated xhtml files. Typically, '.html' or '.htm' can be used as the extension name for the generated xhtml files. You can also specify other extension name. The default is '.html'.	No
args.css	User specified css file, it can be a local file or remote file in the web. **Note:** It must be a filepath relative to URL or local root dir base on the type of `args.csspath`.	No
args.csspath	The path for css reference, it can be a URL start with 'http://' or 'https://', it also can be a local absolute directory. Default will be output directory. **Note:** `args.csspath` should end with '/' if it is a URL, or file separator for local path.	No
args.hdf	The name of the file containing XHTML to be placed in the HEAD area.	No
args.hdr	The name of the file containing XHTML to be placed in the BODY running-heading area.	No
args.ftr	The name of the file containing XHTML to be placed in the BODY running-footing area.	No

targets in *conductor.xml*

The following targets in *conductor.xml* will call the complete processing of DITA files which can be loaded by users.

dita2docbook
Transform DITA topic or DITA map into docbook output.

dita2eclipsehelp
Transform DITA topic or DITA map into eclipse help plugin based on xhtml.

Table1. Parameter Table of dita2eclipsehelp

Parameter	Description	Required
args.eclipsehelp	The root file name of the output eclipsehelp toc file in eclipsehelp transformation. The default is the name of input ditamap file.	No

dita2eclipsecontent
Transform DITA topic or DITA map into eclipse content provider based on xhtml.

Table1. Parameter Table of dita2eclipsecontent

Parameter	Description	Required
args.eclipsecont	The root file name of the output eclipse content provider toc file in eclipsecontent transformation. The default is the name of input ditamap file.	No

dita2htmlhelp
Transform DITA topic or DITA map into html help output based on html.

dita2javahelp
Transform DITA topic or DITA map into java help output based on html.

Table1. Parameter Table of dita2javahelp

Parameter	Description	Required
args.javahelp.toc	The root file name of the output javahelp toc file in javahelp transformation. The default is the name of input ditamap file.	No
args.javahelp.ma	The root file name of the output javahelp map file in javahelp transformation. The default is the name of input ditamap file.	No

dita2xhtml
Transform DITA topic or DITA map into xhtml web output.

Table1. Parameter Table of dita2xhtml

Parameter	Description	Required
args.xhtml.toc	The root file name of the output xhtml toc file in xhtml transformation. The default is 'index'.	No

dita2pdf
Transform DITA topic or DITA map into pdf.

Table1. Parameter Table of dita2pdf

Parameter	Description	Required
args.fo.img.ext	The extension name of image file in pdf transformation. Only '.jpg', '.gif' are valid value. The default is '.jpg'. **Note:** Only one extension supported in the same transformation, image files with other extensions will be renamed to the specified extension.	No

dita2troff

Transform DITA map into troff, which is the system menu style in UNIX system.

pretargets.xml

The Ant script file which contains all targets for pre-process.

ditatargets.xml

The Ant script file which contains all targets for main transformation.

catalog-ant.xml

The xml catalog information which is used by Ant.

dita2troff

Transform DITA topic or DITA map into eclipse help plugin based on xhtml.

Sample ant script

These ant scripts are in `/ant/` directory. They are simple and easy to learn. From these files, you can learn how to write your own Ant script to build your own process.

Here is a sample template for writing an Ant script that executes transformation to xhtml in `/ant/` directory

```
<?xml version="1.0"
      encoding="UTF-8" ?> <project
      name="sample_xhtml" default="all"
      basedir=".."> <import
      file="${basedir}${file.separator}conductor.xml"/>
      <property name="dita.extname" value=".xml"/>
      <target name="all" depends="sample2xhtml">
      </target> <!-- revise below here --> <target
      name="sample2xhtml" depends="use-init">
      <antcall target="dita2xhtml"> <param
      name="args.input" value="@DITA.INPUT@"/> <param
      name="output.dir" value="@OUTPUT.DIR@"/>
      </antcall> </target> </project>
```

After you write the input file and output directory to overwrite @DITA.INPUT@ and @OUTPUT.DIR@, the script can execute the transformation from your input to xhtml by this command. The property of dita.extname is a global variable with which you can set the file extension name of the topic file. The default dita.extname is "*.xml*" . You can also set it to "*.dita*" according to OASIS DITA recommendation.

```
c:\pkg\DITA-OT1.0>ant -f ant/template_xhtml.xml
```

All of targets we use here are defined in *conductor.xml*. Therefore, you need to import that file before calling the target.

Building DITA output with Java command line

The DITA Open Toolkit release 1.0.2 or above provides a command line interface as an alternative for users with little knowledge of Ant to use the toolkit easily.

Running example

1. Change into the DITA Open Toolkit installation directory.
2. On the command line, enter the following command:

   ```
   java -jar lib/dost.jar /i:samples/sequence.ditamap /outdir:out
   /transtype:xhtml
   ```

This particular example creates a properties file, and then calls Ant using those properties to build the sample *sequence.ditamap* file and outputs the xhtml results to the out directory. You can add other parameters to the property file. See the following *Table of supported parameters* for details.

Note:

1. In this example, the '/' symbol preceded by a space is the separator for each parameter.
2. Currently, the parameters */filter*, */ftr*, */hdr*, and */hdf* require an absolute path.
3. The properties file is saved in the ${dita.temp.dir} directory. This properties file can also be used in future transform. The following command provides an example:

```
ant -f conductor.xml -propertyfile
${dita.temp.dir}/property.temp
```

Supported parameters

Table of supported parameters lists the supported parameters (their Ant names are within the braces) that you can set with this tool.

Table1. Table of supported parameters

Parameter	Description
/i:{args.input}	The path and name of the input file. This argument should be in the same upper or lower case with the filename on file system. **Note:** This parameter must be provided if `dita.input` and `dita.input.dirname` not be provided.
/if:{dita.input}	The name of the input file . **Note:** This parameter must be provided if `args.input` not be provided. And this parameter must be used together with the `dita.input.dirname` parameter. The result of this combination is equivalent to use only the `args.input` parameter. It is an alternative way to specify the path and name of the input file.
/id:{dita.input.dirname}	The input directory which contains the input file. **Note:** This parameter must be provided if `args.input` not be provided. And this parameter must be used together with the `dita.input` parameter. The result of this combination is equivalent to use only the `args.input` parameter. It is an alternative way to specify the path and name of the input file.
/outdir:{output.dir}	The path of the output directory.
/tempdir:{dita.temp.dir}	The directory of the temporary files. The default is 'temp'.
/ditaext:{dita.extname}	The file extension name of the input topic files, for example, '.xml' or '.dita'. The default is '.xml'.
/transtype:{transtype}	The transformation type. Currently, the supported values include xhtml, pdf, javahelp, eclipsehelp, htmlhelp, eclipsecontent, troff, and docbook.
/filter:{dita.input.valfile}	The name of the file containing *filter/flagging/revision* information.
/draft:{args.draft}	Default "hide draft & cleanup content" processing parameter (“no” = hide them); “no” and “yes” are valid values; non- “yes” is ignored.
/artlbl:{args.artlbl}	Default "output artwork filenames" processing parameter; “no” and “yes” are valid values; non-“yes” is ignored.
/ftr:{args.ftr}	The name of the file containing XHTML to be placed in the BODY running-footing area.
/hdr:{args.hdr}	The name of the file containing XHTML to be placed in the BODY running-heading area.

Parameter	Description
/hdf:{args.hdf}	The name of the file containing XHTML to be placed in the HEAD area.
/csspath:{args.csspath}	The path for css reference, it can be a URL start with 'http://' or 'https://', it also can be a local absolute directory. Default will be output directory. **Note:** `args.csspath` should end with '/' if it is a URL, or file separator for local path.
/css:{args.css}	User specified css file, it can be a local file or remote file in the web. **Note:** It must be a filepath relative to URL or local root dir base on the type of `args.csspath`.
/copycss:{args.copycss}	The parameter to specify whether copy user specified css files to the directory specified by `args.csspath`. Only "no" and "yes" are valid values.
/indexshow:{args.indexshow}	The parameter to specify whether each index entry should display within the body of the text itself. Only "no" and "yes" are valid values.
/outext:{args.outext}	The output file extension name for generated xhtml files. Typically, '.html' or '.htm' can be used as the extension name for the generated xhtml files. You can also specify other extension name. The default is '.html'.
/xsl:{args.xsl}	The xsl file to replace the default xsl file. It will replace dita2docbook.xsl in docbook transformation, dita2fo-shell.xsl in pdf transformation, dita2xhtml.xsl in xhtml/eclipsehelp transformation, and dita2html.xsl in javahelp/htmlhelp transformation.
/cleantemp:{clean.temp}	The parameter to specify whether to clean the temp directory before each build. Only "no" and "yes" are valid values. The default is yes.
/foimgext:{args.fo.img.ext}	The extension name of image file in pdf transformation. Only '.jpg', '.gif' are valid value. The default is '.jpg'. **Note:** Only one extension supported in the same transformation, image files with other extensions will be renamed to the specified extension.
/javahelptoc:{args.javahelp.toc}	The root file name of the output javahelp toc file in javahelp transformation. The default is the name of input ditamap file.
/javahelpmap:{args.javahelp.map}	The root file name of the output javahelp map file in javahelp transformation. The default is the name of input ditamap file.
/eclipsehelptoc:{args.eclipsehelp.toc}	The root file name of the output eclipsehelp toc file in eclipsehelp transformation. The default is the name of input ditamap file.
/eclipsecontenttoc:{args.eclipsecontent.toc}	The root file name of the output eclipse content provider toc file in eclipsecontent transformation. The default is the name of input ditamap file.
/xhtmltoc:{args.xhtml.toc}	The root file name of the output xhtml toc file in xhtml transformation. The default is 'index'.

Controls, parameters, tweaks, and gizmos for *dita2htmlImpl.xsl*

If the available methods can not fully match your own output requirements, DITA Toolkit supports other ways to customize or enhance the transforms without having to modify core transforms directly.

The *dita2htmlImpl.xsl* file is the main XHTML processor to produce the output. You can work with the following variables and parameters to change the way of processing.

If you need to make code changes to *dita2htmlImpl.xsl* to change a variable value, the preferred mechanism is to create an override transform and place your changed code in the new transform.

Here is an example of an override transform:

1. In the `/xsl/` directory, make a copy of the *dita2xhtml.xsl* file and change the filename of the copy.
2. Add your XSLT changes within the new file.

Note:

- Making editing changes to XSLT transforms is not included in this document; refer to another XSLT reference for guidance.
- It is not recommended to edit any of the DITA distribution files, because your modification might be erased by package updates.
- You are responsible for any change you make to DITA transforms. If you need help, the DITA forum on developerWorks may be a useful resource.

Global variable declarations

If you want to change the values of the following global variable declarations to meet your output requirements, copy the appropriate XSL directive into an override stylesheet that imports the *dita2htmlImpl.xsl* file and make your editing changes in the stylesheet.

Variable name	Explanation	Default Value
afill	Filler for A-name anchors (link-to points that have no data content in and of themselves; some browsers fail to link if a named anchor has no text)	null string (could be a space character, , , etc.)

For example, copy the **afill** variable declaration into your override stylesheet and change the content to represent the actual copyright owner of your content (this string will be copied into the result HTML document as a comment):

```
<xsl:variable
        name="afill"> </xsl:variable>
```

Default values for externally modifiable parameters

This topic lists the default values for externally modifiable parameters.

These default values can be changed at run time by using the parameter-passing syntax of your XSLT processor (if it supports command-line parameters). If your processor does not support parameter-passing on the command line, copy the XSL directives you wish to change into an override XSLT stylesheet, change the values as needed, and import *dita2htmlImpl.xsl* at the top of this new stylesheet. *dita2xhtml.xsl* is an example of an override stylesheet that you can copy and modify as needed.

Parameter name	Explanation	Default value
dita-css	Default CSS filename parameter, usually the name of your site's overall stylesheet.	`commonltr.css`
bidi-dita-css	Default CSS filename parameter for bi-direction language, usually the name of your site's overall stylesheet.	`commonrtl.css`
CSS	User's CSS filename parameter. This can be the name of a stylesheet used by one or more topics within an overall group. This stylesheet can use the CSS cascade effect to modify existing properties or it can override or define new properties.	`null`
CSSPATH	Default CSS path parameter. This specifies a path for the cascading style sheet (CSS). This allows you to place the CSS in one place and have several different topics point to it. If no CSSPATH is specified, the CSS is assumed to be in the same directory as the XHTML.	`null`
HDF	The name of the file which contains XHTML codes to be placed in the HEAD area.	`null`
HDR	The name of the file which contains XHTML codes to be placed in the BODY running-heading area.	`null`
FTR	The name of the file which contains XHTML codes to be placed in the BODY running-footing area.	`null`
ARTLBL	Default **output artwork filenames** processing parameter; `no` and `yes` are valid values; any other value is ignored.	`no`
DRAFT	Default **hide draft & cleanup content** processing parameter (`no`= hide them); `no` and `yes` are valid values; any other value is ignored.	`no`
INDEXSHOW	Default **hide index entries** processing parameter (`no` = hide them); `no` and `yes` are valid values; any other value is ignored.	`no`
YEAR	The year for the copyright.	`2005l`
OUTEXT	Default **output extension** processing parameter; `htm` and `html` are valid values.	`html`
WORKDIR	The working directory, relative to the stylesheet, that contains the document being transformed. Needed as a directory prefix for the @conref and @href document() function calls.	`./`
PATH2PROJ	The path back to the project. Used for *c.gif*, *delta.gif*, and *.css* files to allow users to have these files in 1 location.	`null`
FILENAME	The file name (file name and extension only - no path) of the document being transformed. Needed to help form debugging messages. **Note:** This value is not inherent to the XSLT processor; typically, when the transform starts, the input filename will be passed to the processor's command line as a parameter. Any resulting debugging messages will echo the file name.	`null`
FILTERFILE	The name of the file that contains filter/flagging/revision information.	`null`
DBG	Debug mode which enables XSL debugging XSL messages. Needed to help form debugging messages. `no` and `yes` are valid values; any other value is ignored.	`no`

DITAEXT	DITAEXT file extension name of dita topic file.	null

For example, the following sample invocation shows how to turn on draft mode using the Saxon XSLT processor:

```
c:pkg\dita12\doc>java -jar <saxon_dir>/saxon.jar abc.htm
        dita-tweaks.xml ..\xsl\dita2htmlImpl.xsl DRAFT=yes
```

The effect of this parameter will be to show the content of all <draft-comment> and <required-cleanup> elements with highly visible styling for use by reviewers.

Note: To invoke a process using parameters, please check the documentation for your XSLT processor. Most current XSLT 1.0 processors support a non-standard command line interface for parameters.
Note: Parametric tweaks cannot be applied from internal stylesheet links (that is, the <?xml-stylesheet ...?> processing instruction) as such associations do not provide a way to pass parameters, even if a browser-specific renderer is capable of using such data. To cause a browser-based view to show something ordinarily affected by a command-line parameter, such as the DRAFT="yes" effect, embed the alternative value directly in the override stylesheet that is named in the stylesheet processing instruction:

```
<xsl:param name="DRAFT"
        select="'yes'"/>
```

Stubs for user-provided override extensions

The *dita2htmlImpl.xsl* stylesheet provides code stubs that extend the appearance of your HTML result document. If you copy these stubs into your override stylesheet and provide your own code within them, the result content will be pasted into appropriate parts of the overall HTML page.

Regions that can be modified by these stubs include header, footer, topic-top blurbs (a common location for mini-contents boxes), self-contained scripts (such as JavaScript used commonly for DHTML support), self-contained styles, flagging based on property value matches, panel titles and prefixes for panel titles (to auto-generate explicit bookmarks for your users). See the section of *dita2htmlImpl.xsl* called **start of override stubs** for the examples of mostly empty stubs that you can modify.

For example, copy the following template rule into an override file, such as a copy of *dita2xhtml.xsl*, to generate a mini table of contents at the top of a topic that directly nests child topics:

```
<!-- override for main stub --> <xsl:template
       name="gen-user-sidetoc"> <!-- if there are nested
       topics... --> <xsl:if
       test="descendant::*[contains(@class,' topic/topic
       ')]"> <p> <table width="150"
       align="right" border="1" frame="box"
       rules="none"> <tr><td height="5"
       bgcolor="#0033CC" align="center">
       <b><font
       color="#FFFFFF">Contents:</font></b></td>
       </tr> <xsl:for-each
       select="descendant::*[contains(@class,' topic/topic
       ')]"> <xsl:variable
       name="ttext"><xsl:value-of
       select="*[contains(@class,' topic/title
       ')]"/></xsl:variable> <tr><td
       class="toc">- <a
       href="#{generate-id()}"><xsl:value-of
       select="$ttext"/></a> <!--recursive call for
       subtopics here"/--> </td></tr>
       </xsl:for-each> </table> </p> </xsl:if>
       </xsl:template>
```

Remember: Do your modifications on the copies of stylesheets so that you can turn back on the original!