DITA Readme map

[vertical list of authors]

[cover art/text goes here]

Contents

Installing DITA OT 1.3

The software that the previous releases of DITA OT depend on are redistributed. The installation package of DITA OT 1.3 includes the software, so the configuration process is streamlined.

Before installing DITA Open Toolkit 1.3, you need to complete the following steps:

- Download and configure JDK properly.
- (Optional) Download and configure HTMLHelp Compiler properly for HTMLHelp transformation.
- (Optional) Download and configure JavaHelp Compiler properly for JavaHelp transformation.

Then, you need to complete the following steps:

1. Download the full DITA OT 1.3 package.
2. Run the batch file "`startcmd.bat`" for Windows or "`startcmd.sh`" for Linux to set up the necessary environment variables.
3. Run the transformation in the command-line window by using Java command line or ANT as you did in the previous releases of DITA OT.

You can still use the small package to install DITA OT 1.3:

- Download the small package.
- Configure the system environment as you did in previous releases of DITA OT, but add two more paths %DITA_HOME%\lib;%DITA_HOME%\lib\resolver.jar; to the CLASSPATH parameter.
- Run the package by using Java command line or Ant.

Enhanced command line help

In DITA OT 1.3, the command line help function is enhanced to improve usability. You can know the version of toolkit and the usage of the command line from the command line help by using the following commands:

```
java -jar lib/dost.jar -version

java -jar lib/dost.jar -h
```

You can see the brief description of the supported parameters in the command line window when you type a specific command. For example, if you type `java -jar lib/dost.jar -h` , you can get the following result:

```
D:\DITA-OT1.3_fullpackage_bin\DITA-OT1.3>java -jar lib/dost.jar -h
java -jar lib/dost.jar [mandatory parameters] [options]
Mandatory parameters:
  /i:{args.input}        specify the input file
  /transtype:{transtype} specify the transformation type
Options:
  -help, -h              print this message
  -version               print the version information and exit
  /basedir:{basedir}     specify the working directory
  /ditadir:{dita.dir}    specify the toolkit's home directory
  /outdir:{output.dir}   specify the output directory
  /tempdir:{dita.temp.dir} specify the temporary directory
  /logdir:{args.logdir}  specify the log directory
  /ditaext:{dita.extname} specify the dita file extension
  /filter:{dita.input.valfile} specify the filter file
  /draft:{args.draft}    specify whether to output draft info
  /artlbl:{args.artlbl}  specify whether to output artwork filenames
  /ftr:{args.ftr}        specify the file to be placed in the BODY
running-footi
ng area
  /hdr:{args.hdr}        specify the file to be placed in the BODY
running-headi
ng area
  /hdf:{args.hdf}        specify the file to be placed in the HEAD area
  /csspath:{args.csspath} specify the path for css reference
  /css:{args.css}        specify user css file
  /cssroot:{args.cssroot} specify the root directory for user specified
css file

  /copycss:{args.copycss} specify whether to copy user specified css
files
  /indexshow:{args.indexshow} specify whether each index entry should
display wi
thin the body of the text itself
  /outext:{args.outext}  specify the output file extension for generated
xhtml f
iles
  /xsl:{args.xsl}            specify the xsl file used to replace the
default xs
l file
  /cleantemp:{clean.temp} specify whether to clean the temp directory
before eac
h build
  /foimgext:{args.fo.img.ext} specify the extension of image file in pdf
transfo
rmation
  /javahelptoc:{args.javahelp.toc} specify the root file name of the
output java
help toc file in javahelp transformation
  /javahelpmap:{args.javahelp.map} specify the root file name of the
output java
help map file in javahelp transformation
  /eclipsehelptoc:{args.eclipsehelp.toc} specify the root file name of
the outpu
t eclipsehelp toc file in eclipsehelp transformation
  /eclipsecontenttoc:{args.eclipsecontent.toc} specify the root file name
of the
 output Eclipse content provider toc file in eclipsecontent
transformation
  /provider:{args.eclipse.provider} specify the provider name of the
eclipse hel
p output
  /version:{args.eclipse.version} specify the version number of the
```

```
eclipse help
 output
  /xhtmltoc:{args.xhtml.toc} specify the root file name of the output
xhtml toc
file in xhtml transformation
  /ditalocale:{args.dita.locale} specify the locale used for sorting
indexterms.

  /fooutputrellinks:{args.fo.output.rel.links} specify whether to output
related
 links in pdf transformation
  /fouserconfig:{args.fo.userconfig} specify the user configuration file
for FOP

  /htmlhelpincludefile:{args.htmlhelp.includefile} specify the file that
need to
 be included by the HTMLHelp output
```

Extensible metadata attributes

OASIS DITA 1.1 provides the DITA architects with an enhanced feature, extensible metadata attributes. If the architects want to achieve multiple purposes in one attribute, especially in a selective attribute, they can use the extensible metadata attributes.

Note:

- Because OASIS DITA 1.1 is not yet an approved standard as of the release of DITA OT 1.3, the functionality described here should be considered a *preview* capability.
- The specification and the defined functions that need to be supported can change by the time OASIS formally approves DITA 1.1.

Example

The following example illustrates how people of different roles use the extensible metadata attributes in DITA 1.1.

- As a DITA architect of a team, you can perform the following actions:
  - #. Define new attributes that the team needs, for example, "proglanguage".
  - #. Express each new attribute as a separate domain package, for example, proglanguage.mod, with the new attribute specialized from the "props" attribute.
  - #. Integrate the domain packages into the authoring DTDs or schemas:
    1. Redefine the "props" attribute entity to include the "proglanguage" attribute. Similarly, you can redefine element entities to integrate new domain elements.
    2. Add the new attribute domain to the list of domains in the domains attribute, preceded by an "a", for example, `domains="a(props proglanguage)"`.
- As an author, you can perform the following actions:
  - #. Add values to the new attributes of an element.
  - #. Define values in the DITA filter file.
  - #. Transform the DITA source files to remove or flag the content based on the new attributes, for example, flagging all `proglanguage="Java"`

  After you perform these actions, another user can reuse the content.

  A specialization-unaware trademarking tool requires generalization of the contributed content. If the user runs all the content through the tool, the content is processed and filtered against the new attributes after the generalization. The new attributes are now collapsed into the "props" attribute.
  - #. The generalization turned `proglanguage="Java"` into `props="proglanguage(Java)"`.
  - #. The conditional processing transform recognizes the new form as equivalent to the old, and the instruction "`flag all proglanguage=java`" operates on either `props="proglanguage(Java)"` or `proglanguage="Java"`.

Graphic scaling improvement

Graphic scaling improvement is an enhanced feature that DITA Open Toolkit 1.3 provides. DITA OT 1.3 supports this feature in the transformation for different outputs, such as HTML, XHTML, PDF, and FO. This feature is not applicable in RTF output.

Note:

- Because OASIS DITA 1.1 is not yet an approved standard as of the release of DITA OT 1.3, the functionality described here should be considered a *preview* capability.
- The specification and the defined functions that need to be supported can change by the time OASIS formally approves DITA 1.1.

To implement this feature, you must first meet the following prerequisites:

- Install and configure the DITA Open Toolkit 1.3 successfully.
- Ensure that the image file referred to by the `<image>` tag exists.

In DITA 1.1, there are some attributes that you can use to set the actual display size of the pictures in the `<image>` tag, such as "width", "height", and so on.

You can set the actual display size of the image in the output by taking the following steps:

1. Specify the height and width of the picture in the "height" and "width" attributes of the `<image>` tag, for example, `<image height="80" width="60" href="a.jpg"/>`
2. (Optional) Specify the metric of the length in the height and width attributes fields, for example, `<image height="80pc" width="60pc" href="a.jpg"/>`. The metrics currently supported are: px, pc, pt, in, cm, mm, em. The default is px. **Note:** If you do not specify the metric of the length, the toolkit will use the default metric, px.
3. Run the transformation to generate the outputs, such as xhtml, HTML, and FO, that support graphic scaling.

In the final output, you can see the image displayed in the size that you expected. As in this example, the picture will be displayed by 80 pt in height and 60 pt in width.

You can also use the scaling function in setting the actual display size of the image in the output by taking the following steps:

1. Specify the height and width of the picture in the "height" and "width" attributes of the `<image>` tag, and the metric of the length.
2. Specify the scale rate in the scale attribute after you specify the height and width for the image, for example, `<image height="80pc" width="60pc" href="a.jpg" scale="0.8"/>`. Scale="0.8" means the picture in the output will be displayed at 80% of the size that you specified by height and width.
3. Run the transformation to generate the outputs that support image scaling, such as xhtml, HTML, and FO.

In the final output, you can see the image displayed in the size that you expected. As in this example, the picture will be displayed by 64 pt in height and 48 pt in width.

Indexing

DITA 1.1 supports the following new indexing elements:

- <index-see>
- <index-see-also>
- <index-sort-as>
- <index-range-start>
- <index-range-end>

Note:

- Because OASIS DITA 1.1 is not yet an approved standard as of the release of DITA OT 1.3, the functionality described here should be considered a *preview* capability.
- The specification and the defined functions that need to be supported can change by the time OASIS formally approves DITA 1.1.

See and See Also indexing elements

In DITA 1.0, you cannot specify the <see> and <see also> index entries by using the current <indexterm> element. The DITA1.1 standard introduces the following new child elements for <indexterm> that support this functionality:

- index-see
- index-see-also

For example, you can add an index entry, as illustrated in the following text in the DITA source file:

```
<indexterm>computer
   <index-see>monitor</index-see>
   <index-see-also>Illustration</index-see-also>
</indexterm>
```

Then, if you generate a PDF output with the indexing function enabled, you can see the following index entries in the PDF output:

```
computer 43
        See monitor
        See also Illustration
```

The "monitor" and "Illustration" entries after "see" and "see also" will not be links to the "monitor" and "Illustration" index entries in a PDF output.

Index entries will only be processed when you generate HTMLHelp and JavaHelp. For HTMLHelp and JavaHelp, the index contains an entry that uses the text "See xxx" or "See also xxx". The "See xxx" or "See also xxx" index entries will link to their parent index term.

Note:

- For HTML output, indexing is ignored.
- For PDF output, you must enable indexing using the FO plugin provided by Idiom.

For example, if you put the following content in the source file,

```
<indexterm>computer
   <index-see>monitor</index-see>
</indexterm>
```

the output is as follows:

```
computer
  See monitor
```

Sort order indexing elements

With the DITA 1.1 standard, you can specify a sort phrase and sort index entries under the sort phrase. This feature provides you with the flexibility to sort an index entry in a different way. Typically you can disregard insignificant leading text, such as punctuation or words like "the" or "a". If you want to sort <data> under the letter D rather than the character "<", you can include such an entry under both the punctuation heading and the letter D. Thus, there can be two index entry directives differentiated only by the sort order.

For example, if you put the following content in the source file,

```
  <indexterm>data<index-sort-as>key</index-sort-as></indexterm>
  <indexterm>indextest<index-sort-as>abc</index-sort-as></indexterm>
```

the output should be:
indextest data

If you have written an XML book with many punctuation-laden entries in its index, you can use the <index-sort-as> element to specify how the sorting method of the entries if the punctuation marks are eliminated. For example, `<data>` is always displayed as an entry <data> in the index term under the letter D; otherwise, all the entries with punctuations will be sorted under "<".

Here is another example. In a translation project, a document needs to be translated into Japanese. Many of the index entries contain kanji, which need to be sorted in phonetic order. The translators, who can understand the language and see the entry in its context, can insert the `<index-sort-as>` elements into the `<indexterm>` elements as part of their localization work.

Page-range indexing elements

In DITA OT 1.3, you can indicate page ranges instead of individual references over consecutive pages. Page ranges indicate where the index entry links to an extended discussion that goes over a number of pages. This is typically manifested as a page range like 34-36. This is distinguished from individual references over consecutive pages (34, 35, 36). The page-range indexing function is enabled when you use the FO plugin.

For example, you can add a page spanning index entry:

```
<indexterm>DITA<index-range-start/></indexterm>
```

. Later in the same topic, you can add a range terminating marker: <indexterm>DITA<index-range-end/></indexterm>. This spans 4 pages on the paper, as illustrated in the following example.

```
DITA, 46-49
```

Note: If you generate HTMLHelp, JavaHelp, and XHTML outputs, the page-range indexing elements are ignored.

Supporting ICU in index sorting

With enabled ICU interface, DITA Open Toolkit 1.3 helps you get correctly sorted index output for different languages.

During normal transformation, the toolkit tries to find if there are ICU classes inside the `classpath` element. If ICU exists, the toolkit uses ICU's Collator class to do the comparing and sorting work. If no ICU is found, the toolkit will use JDK's Collator class to do the comparing and sorting work. ICU is packed in the big package in DITA OT 1.3

New element <abstract>

You can now use a new element <abstract> in DITA topics. The <abstract> element can include complex markups besides the <shortdesc> element. You can put the <shortdesc> element inside the <abstract> element, together with many other elements. The following examples illustrate how you can use the <abstract> element..

If you use several <shortdesc> elements inside the <abstract> element, they will be concatenated when pulled for hover help. After you format the source files, the content inside the <abstract> element will be transformed into normal text.

Note:

- Because OASIS DITA 1.1 is not yet an approved standard as of the release of DITA OT 1.3, the functionality described here should be considered a *preview* capability.
- The specification and the defined functions that need to be supported can change by the time OASIS formally approves DITA 1.1.

Examples

Example 1

In DITA 1.0, you can only use the <shortdesc> element that cannot contain the <p> element.

```
 <shortdesc>This is a short description in DITA 1.0. It <b>cannot</b>
contain paragraphs.</shortdesc>
```

Example 2

This example illustrates how you can use different elements besides <shortdesc> inside the <abstract> element, and apply different styles to the text inside the <abstract> element.

```
    <abstract>
        <shortdesc>This is the short description</shortdesc>
        <ol>
            <li>This is a <i>list</i>.</li>
        </ol>
        <p>This is a <b>paragraph</b>.</p>
        <codeblock>Here are some codes.</codeblock>
        <filepath>This is the file path.</filepath>
    </abstract>
```

Example 3

This example illustrates how you can use both the <shortdesc> element and plain text inside the <abstract> element.

```
 <abstract><shortdesc>This topic is about short description.</shortdesc>.
Short description is very important, so read more.</abstract>
```

New element <data>

In DITA 1.1, you can use new element, <data>. This element and the content inside it is ignored in the transformation process of DITA files.

Note:

- Because OASIS DITA 1.1 is not yet an approved standard as of the release of DITA OT 1.3, the functionality described here should be considered a *preview* capability.
- The specification and the defined functions that need to be supported can change by the time OASIS formally approves DITA 1.1.

As an author, when you create DITA files, you can add the <data> element, and put content inside it. When you transform the DITA files to the output that you want, the transformation ignores the <data> element and any content inside.

As a specializer, when you specialize the <data> element, and put information inside the specialized element, you can create a transform override to use the information.

Supporting two file extensions in one DITA map

DITA Open Toolkit supports two different file extensions, ".dita" and ".xml". Previous releases of DITA Open Toolkit do not support the transformation of DITA maps containing inconsistent file types, such as one DITA map containing both .dita and .xml files. Though you can create either .dita or .xml files, you cannot include both kinds of files in one DITA map. This makes file reuse difficult, because you have to change the file extensions manually make them consistent in one DITA map.

In DITA OT 1.3, you can include both .xml and .dita as the file extensions in one DITA map and transform the DITA map into your desired output without manually changing the file extensions.

If you include both .xml and .dita files in one DITA map, and specify /ditaext:.dita in Java command, the .xml files are transferred to .dita files and put in the temp directory together with the .dita files. If you specify /ditaext:.xml in Java command, all the .dita files are transferred to .xml files under the temp directory. The default process option is changing all files into .xml files.

Note:

- It is not suggested that you include files with the same root name but different extensions in the same directory because this might cause unexpected problems.
- Error messages and warning messages in the console might not reflect the real extension. For example, if there is an incorrect usage in a.dita, the warning message in the console might refer to a.xml, because a.dita was changed into a.xml in the temp directory.

You might use other file extensions together with .dita and .xml in one ditamap as well, such as .dit, but they are not tested in DITA OT 1.3 and thus you might take the risk of transformation failure.

Supporting foreign content vocabulary

In DITA 1.1, you can use the <unknown> element to incorporate existing standard vocabularies for special content, like MathML and SVG, as inline objects.

Note:

- Because OASIS DITA 1.1 is not yet an approved standard as of the release of DITA OT 1.3, the functionality described here should be considered a *preview* capability.
- The specification and the defined functions that need to be supported can change by the time OASIS formally approves DITA 1.1.

As an author, when you create DITA files, you can add the <unknown> element, and put content inside it. The <unknown> element and any content inside it is ignored when you transform the DITA files to your desired output.

As a specializer, when you specialize the <unknown> element, and then put information inside the specialized element, you can create a transform override that allows the information to appear correctly in the output.

Refactored ANT tasks

The ANT tasks are refactored so that developers can easily find the targets they need and make their own extensions. Developers can still use Java Command Line as before. Ant Build Script "`conductor.xml`" is marked as deprecated, but developers can still use it as before. "`build.xml`" is the renamed version of "`conductor.xml`", and developers can use it as what they do with "`conductor.xml`". New users are suggested to use those separated build scripts like "`build_dita2*.xml`". The original demo ant script "`build.xml`" is renamed to "`build_demo.xml`". "`build_demo.xml`" should be used to verify the functions of the toolkit after this release.

Standard XML catalog resolver

In the previous releases of DITA Open Toolkit, a simple XML catalog resolver is enabled. You do not need to update the reference to dtd in DITA files when the file paths are changed; however, this simple implementation cannot be redistributed because it does not support standard XML catalogs.

In DITA OT 1.3, a standard XML catalog resolver is enabled so that the reference to dtd in DITA files does not need to be updated each time when you change the file paths on your workstation or use another workstation.

With this enhanced feature, when a developer makes a new specialization, the developer only needs to update the mapping between the new dtd file's system id (location relative to the catalog file) and public id (the id assigned by the developer in the head of the DITA or xml file which identifies the corresponding dtd file) in the catalog file (catalog-dita_template.xml), for example, `<public publicId="-//IBM//DTD DITA ABC//EN" uri="dtd/abc.dtd"></public>`.

This enhanced feature does not change the normal behavior of the toolkit.

Topic merge

The topic merge feature improves the build speed of DITA files and reduces the possibility of meeting the out of memory exception in the build process. As illustrated in the following figure, when you run the build in previous releases of DITA Open Toolkit, the build speed is slow and you are likely to get out of memory exception.

With this enhanced topic merge feature, you will be less likely to meet the out of memory exception error when you build output through DITA files. The intermediate merged file will keep the structure information in the DITA map, and the structured toc will be reflected in the output.

To know more about this topic feature, you can write a script file first. DITA OT 1.3 offers a module, `TopicMerge`, that helps you implement this feature. You can use this module to generate the merged files. A sample usage of this module is as follows.

sample.xml:

```
<project name="sample">
        <property name="dita.dir" value="${basedir}"/>
        <import file="${dita.dir}${file.separator}build.xml"/>

        <target name="premerge">
                  <antcall target="preprocess">
                  <param name="args.input" value="${input}"/>
                <param name="output.dir"
value="${dita.dir}${file.separator}output"/>
                </antcall>
        </target>
        <target name="merge"  description="Merge topics"
depends="premerge">
                <basename property="temp.base" file="${input}"
suffix=".ditamap"/>
                <property name="temp.input"
value="${basedir}${file.separator}${dita.temp.dir}${file.separator}${temp.base}"/>
                <dirname property="temp.dir" file="${temp.input}"/>
                <pipeline message="topicmerge" module="TopicMerge"
inputmap="${temp.dir}${file.separator}${temp.base}.ditamap"
extparam="output=${dita.dir}${file.separator}output${file.separator}${temp.base}_merged
      style=${dita.dir}${file.separator}xsl${file.separator}pretty.xsl"
/>
        </target>
</project>
```

Then, you need to type `ant -f sample.xml merge -Dinput="C:\DITA-OT1.3\test.ditamap`" in the command window.
Note: The path for -Dinput must be an absolute path

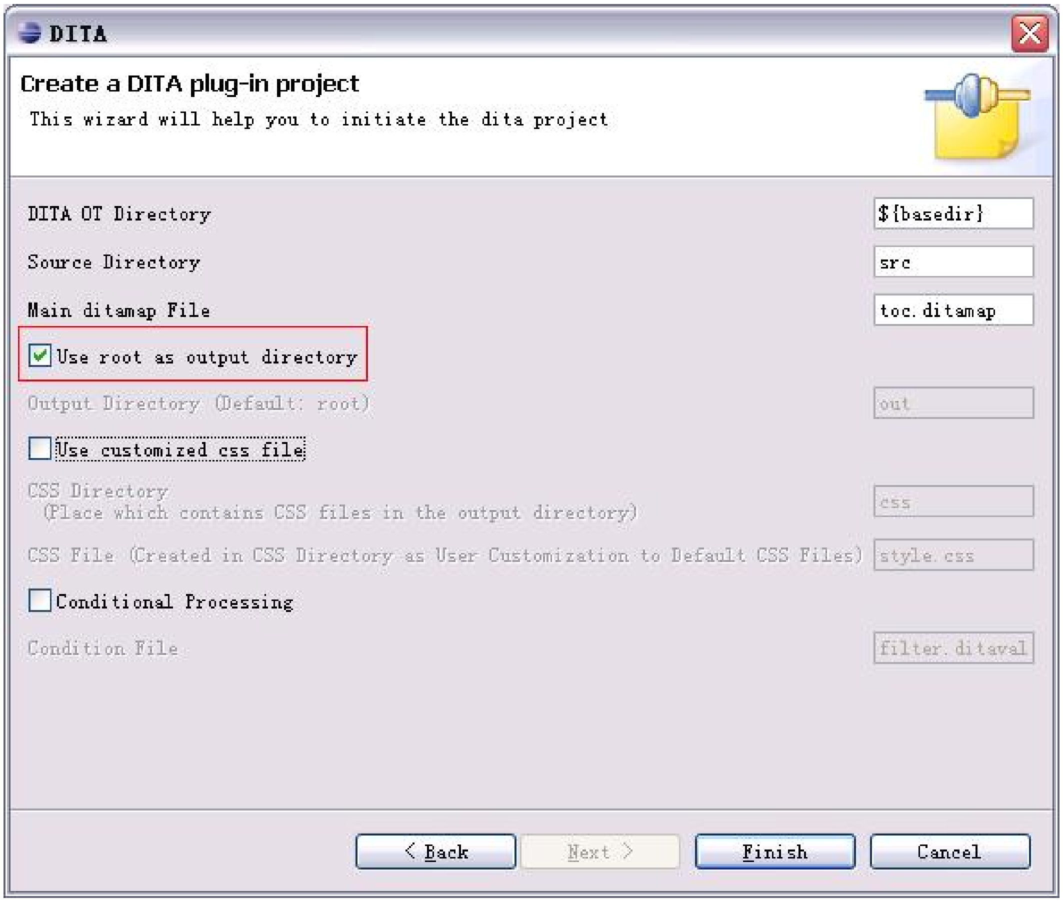
Working with documentation plug-in

You can use a template to develop documentation plug-in with DITA in Eclipse PDE and use DITA OT 1.3 to build and pack the final plug-in. When you want to develop a documentation plug-in with DITA in Eclipse, you cannot use the previous releases of DITA OT in Eclipse to transform DITA to HTML. Though previous releases of DITA OT support the feature to transform DITA files to Eclipse documentation plug-in, they are not integrated with Eclipse. With DITA OT 1.3 integrated with WPT, you can develop document plug-ins with DITA in Eclipse PDE and build and pack the final plug-in by taking the following steps.

1. Create a new PDE project in Eclipse, and apply the DITA template to the project by following the wizard.

2. Set the source directory, the main ditamap file, the output directory (default value is root directory of project), css storage directory (used to contain common.css, commonltr.css, and commonrtl.css), user customized .css file name, and conditional processing ditaval file in the wizard. **Use root as output directory** is selected as the default.

   You can also clear **Use root as output directory** and specify another output directory.



DITA

Create a DITA plug-in project

This wizard will help you to initiate the dita project

DITA OT Directory: ${basedir}

Source Directory: src

Main ditamap File: toc.ditamap

☑ Use root as output directory

Output Directory (Default: root): out

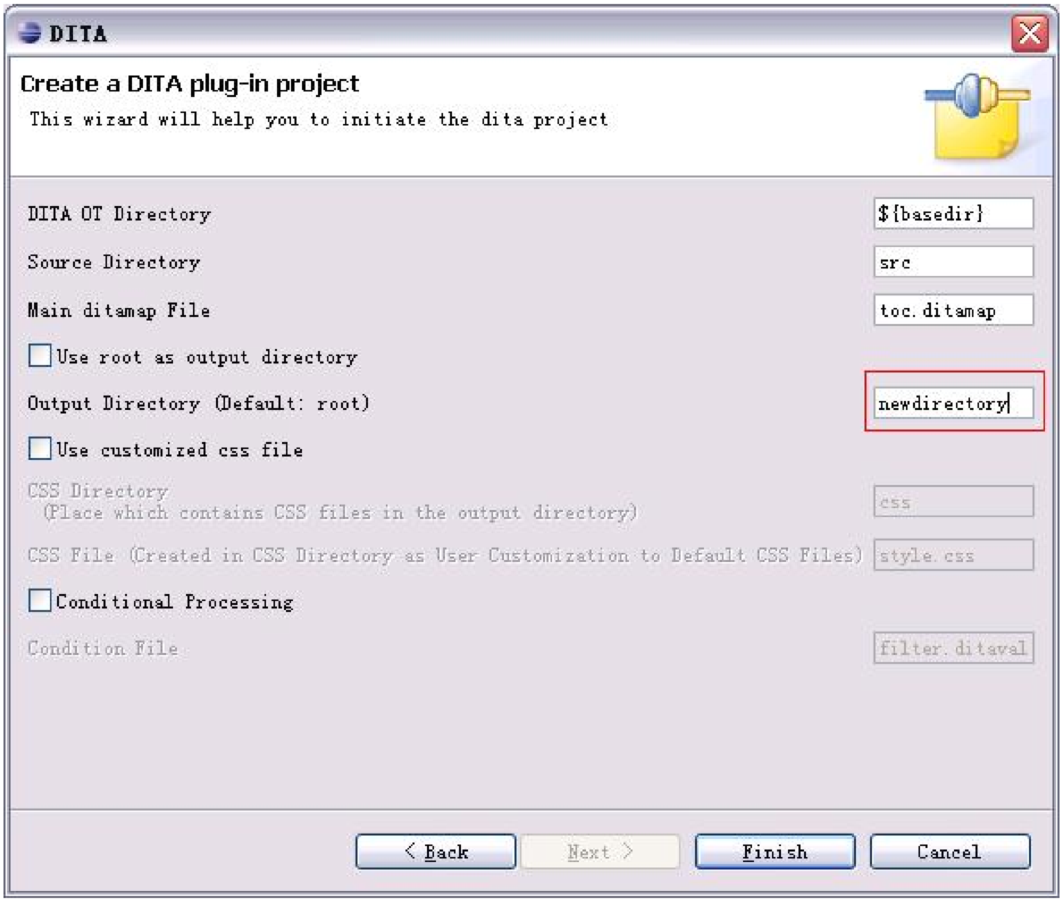
☐ Use customized css file

CSS Directory (Place which contains CSS files in the output directory): css

CSS File (Created in CSS Directory as User Customization to Default CSS Files): style.css

☐ Conditional Processing

Condition File: filter.ditaval



Create a DITA plug-in project

This wizard will help you to initiate the dita project

DITA OT Directory: ${basedir}

Source Directory: src

Main ditamap File: toc.ditamap

☐ Use root as output directory

Output Directory (Default: root): newdirectory

☐ Use customized css file

CSS Directory (Place which contains CSS files in the output directory): css

CSS File (Created in CSS Directory as User Customization to Default CSS Files): style.css

☐ Conditional Processing

Condition File: filter.ditaval

3. Create DITA files in the source directory and a ditamap to include the topic files that you created.
4. Import the DITA files into the `src` directory of the DITA plug-in project you just created.
   #. Right-click a directory that you want to put the imported files and select Import, and then File system.

   #. Select the directory under which you put the DITA files.

   #. Click Finish after you selected the DITA files under the specified directory. The DITA files are then imported to your DITA project.
5. Right click build.xml, select Run As, and then ANT Build.

   Note: If you're using SUN JDK, please download and use the latest Xalan. The Xalan shipped with SUN JDK has some issue that will cause the build failure. You can use the latest Xalan by selecting **ANT Build ...** and include the all of Xalan's jar files in Classpath.

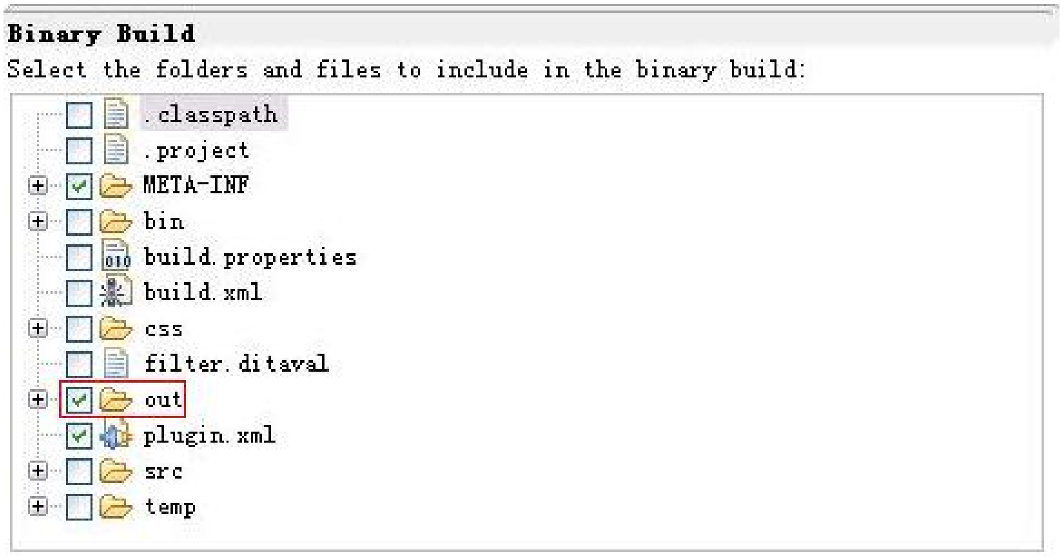
   After the transformation, the output is in the output directory set in build.xml. Refresh the project after the build is successful.

6. Edit the plug-in description of the property file MANITEST.MF in the plug-in editor after you run the ANT build successfully.
    #. Click MANITEST.MF to go to the Overview page.

    #. Edit Dependencies to include org.eclipse.help.

    #. Edit Extensions to add org.eclipse.help.toc; right click the added prgeclipse.help.toc, and select New, and then toc.

    #. Edit the Build Configuration to include the out directory or the directory you specified in Step 2 on page .



Binary Build

Select the folders and files to include in the binary build:

- ☐ .classpath
- ☐ .project
- ☑ META-INF
- ☐ bin
- ☐ build.properties
- ☐ build.xml
- ☐ css
- ☐ filter.ditaval
- ☑ out
- ☑ plugin.xml
- ☐ src
- ☐ temp

    #. Save the changes you made to the property file MANITEST.MF.

7. Export the output to a documentation plug-in.

    Note: build.xml can be customized to meet the requirement of headless build.

    #. Select **File** --> **Export** ; select Deployable plug-ins and fragments and click **Next**.

    #. Select the plug-in you want to export and specify a directory under which you want to put the plug-in package.

    #. Click **Finish** to export the plug-in package.

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 this properties to build the sample *sequence.ditamap* file and outputs the xhtml results to the out directory. You can add other parameters to this properties file. See the following *Table of supported parameters on page 20*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.

Supported parameters

*Table of supported parameters on page 20*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. It is in the same upper or lower case with the filename on file system.
/outdir:{output.dir}	The path of the output directory.
/tempdir:{dita.temp.dir}	The directory of the temporary file. The default is "temp".
/ditaext:{dita.extname}	The file extension name to be used in the temp directory.
/transtype:{transtype}	The transformation type. The supported values include xhtml, pdf, javahelp, eclipsehelp, htmlhelp, eclipsecontent, troff, wordrtf, and docbook.
/filter:{dita.input.valfile}	The absolute file path and the name of the file that contains the *filter/flagging/revision* information.
/draft:{args.draft}	The default "hide draft & cleanup content" processing parameter (“no” = hide them). Only “no” and “yes” are valid values; non- “yes” value is ignored.
/artlbl:{args.artlbl}	The default "output artwork filenames" processing parameter. Only “no” and “yes” are valid values; non- “yes” value is ignored. The default is “no” .
/ftr:{args.ftr}	The absolute file path and the name of the file that contains XHTML to be placed in the BODY running-footing area.
/hdr:{args.hdr}	The absolute file path and the name of the file that contains XHTML to be placed in the BODY

Parameter	Description
	running-heading area.
/hdf:{args.hdf}	The absolute file path and the name of the file that contains 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. The default is output directory. **Note:** The `args.csspath` parameter 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 is a filepath relative to URL or local root dir based on the type of the `args.csspath` parameter.
/cssroot:{args.cssroot}	The root directory of user specified css file. **Note:** If this parameter is set, the ${args.css} should be a filepath relative to args.cssroot.
/copycss:{args.copycss}	The parameter to specify whether to copy user specified css files to the directory specified by the `args.csspath` parameter. Only “no” and “yes” are valid values. The default is “no” .
/indexshow:{args.indexshow}	The parameter to specify whether each index entry needs to display in the body of the text itself. Only “no” and “yes” are valid values. The default is “no” .
/outext:{args.outext}	The output file extension name for generated xhtml files. You can use ".html" or ".htm" 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 replaces dita2docbook.xsl in docbook transformation, dita2fo-shell.xsl in pdf transformation, dita2xhtml.xsl in xhtml/eclipsehelp transformation, dita2rtfImpl.xsl in word transformation, and dita2html.xsl in javahelp/htmlhelp transformation.
/cleantemp:{clean.temp}	The parameter to specify whether to clean the temporary 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. The image files with other extensions are 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 the input ditamap file.

Parameter	Description
/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 the input ditamap file.
/xhtmltoc:{args.xhtml.toc}	The root file name of the output xhtml toc file in xhtml transformation. The default is "index".

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

DITA OT release 1.3.1

Release 1.3.1 is a maintenance release to fix defects and make patches based on release 1.3.

[15 SF Bugs Fixed]

1. SF Bug 1385642 docbook/topic2db.xsl - shortdesc
2. SF Bug 1528638 wordrtf does not correctly number steps
3. SF Bug 1562518 Flag is confusing when a list is mixed with text
4. SF Bug 1563665 Should use CSS to honor rowsep and colsep in table entries
5. SF Bug 1567117 Xref to footnote is not resolved correctly
6. SF Bug 1569671 <reltable> in nested map creates bogus TOC entries
7. SF Bug 1573996 Plugins do not work in plugins directory
8. SF Bug 1574011 Spaces in a file name prevent XHTML output
9. SF Bug 1584186 Bookmap 1.1: <title> element duplicated in mappull
10. SF Bug 1588039 Conref domain checking is sub-par
11. SF Bug 1588624 OT v1.3 map2hhc.xsl error
12. SF Bug 1597444 Java topicmerge breaks when text contains less-than
13. SF Bug 1597473 Nothing references common.css
14. SF Bug 1598109 Java topicmerge does not rewrite image/@href
15. SF Bug 1598230 jhindexer of JavaHelp breaks Search Index for DITAOT content

Note: SourceForge bugs, patches, and RFEs listed above can be found in SourceForge Bugs, Patches, and RFE tracker pages:

- 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 1.3.1

Release 1.3.1 is a maintenance release to fix defects and make patches based on release 1.3.

[15 SF Bugs Fixed]

1. SF Bug 1385642 docbook/topic2db.xsl - shortdesc

2. SF Bug 1528638 wordrtf does not correctly number steps
3. SF Bug 1562518 Flag is confusing when a list is mixed with text
4. SF Bug 1563665 Should use CSS to honor rowsep and colsep in table entries
5. SF Bug 1567117 Xref to footnote is not resolved correctly
6. SF Bug 1569671 <reltable> in nested map creates bogus TOC entries
7. SF Bug 1573996 Plugins do not work in plugins directory
8. SF Bug 1574011 Spaces in a file name prevent XHTML output
9. SF Bug 1584186 Bookmap 1.1: <title> element duplicated in mappull
10. SF Bug 1588039 Conref domain checking is sub-par
11. SF Bug 1588624 OT v1.3 map2hhc.xsl error
12. SF Bug 1597444 Java topicmerge breaks when text contains less-than
13. SF Bug 1597473 Nothing references common.css
14. SF Bug 1598109 Java topicmerge does not rewrite image/@href
15. SF Bug 1598230 jhindexer of JavaHelp breaks Search Index for DITAOT content

DITA OT release 1.3

OASIS DITA 1.1 support

Things to know about OASIS DITA 1.1 support in this release:

1. DITA-OT 1.3 provides preliminary processing support for the upcoming OASIS DITA 1.1 specification (see http://wiki.oasis-open.org/dita/Roadmap_for_DITA_development). Because the proposed OASIS DITA 1.1 DTDs and Schemas are fully backwards compatible with the latest DITA 1.0.1 DTDs and Schemas, the 1.3 Toolkit provides the proposed 1.1 materials as the default DTDs for processing. The XML Catalog resolution maps any references for DITA 1.0 doctypes to the 1.1 DTDs, for example. All processing ordinarily dependent on the 1.0 definition continues to work as usual, and any documents that make use of the newer 1.1-based elements or attributes will be supported with specific new processing function (such as base support for the new <data> element). *Documents created with the proposed OASIS DITA 1.1 DTDs are the only ones ever likely to have features that invoke the specific new 1.1-based processing support*.
   > Important: Because this support is based on a yet-to-be-approved version of the proposed OASIS DITA 1.1 specification, if you choose to investigate any1.1-based function, be aware that the 1.1 implementation in this version of the Toolkit is preliminary and very much forward-looking. Upon final approval of the DITA 1.1 standard, Toolkit developers will, of course, review our implementation to make certain that it conforms to the defined level of reference implementation.
2. Related to the DITA 1.1 preliminary implementation, the much-discussed bookmap updates for DITA 1.1 will be provided as override capabilities for the FO plugin (Idiom's donation). Note that:
   - The FO demo transform code at the 1.2.2 level is still included in the DITA 1.3 package, but is now deprecated.
   - To get the FO updates for 1.3, grab the FO plug-in at its next update, which should be shortly after the 1.3 core Toolkit code is released.
   - The updated FO plug-in will be usable with FOP as well as with XEP.

Changes

The DITA Open Toolkit team understands the need for stability in essential APIs in the Toolit. This verson of the toolkit provides some strategic updates that correct some long-overdue faults in the original implementation. Necessarily, there are some changes to note:

1. Change to build.xml: To make the DITA processing environment more like other Ant-driven build environments, the original build.xml has been renamed as build_demo.xml. The current build.xml in this release is now the normal ANT script entrance for starting a transformation. If you have created Ant tasks that tried to work around the former build.xml architecture, those might need to be revised to

take advantage of the separated function.

2. Change to command line invocations: The "Ant refactoring" exercise for this release has changed some previously documented Ant calls for running demos. This change enables better use of the Ant modules for power users who need to integrate the Toolkit into programming build environments such as Eclipse, but the change affects some documentation. This is a permanent change that should remain stable from now on. Wherever you see an older instruction like "c:\dita-ot>ant all", you now need to indicate the component that contains the demos, so you would type "c:\dita-ot>ant all -f build_demo.xml".
3. Separation of demo targets from formal component targets: Another effect of the Ant refactoring is that the internal programming targets will now be displayed when you type "ant -p". To see both those programmings targets and the demos that are part of this component, type "c:\dita-ot>ant -p -f build_demo.xml". To run just one of the demos that you see in the resulting list, dita.faq for example, type "c:\dita-ot>ant dita.faq -f build_demo.xml".
4. Classpath update to enable catalog resolver: This release now includes the Apache catalog resolver for improved lookup of DTDs by any of the Toolkit components. The fullpackage version of the Toolkit sets up these variables for each session. For the regular (smaller) version of the Toolkit, you need to include lib and lib\resource\resolver.jar into your classpath. For example if your CLASSPATH is like:

   ```
   c:\dita-ot\lib\dost.jar
   ```

   you need to change it to:

   ```
   c:\dita-ot\lib;c:\dita-ot\lib\dost.jar;c:\dita-ot\lib\resolver.jar
   ```

   At any time, the full version can be used like a normal installation as long as you update the system variables either in the evironment settings or in a batch file that sets up the shell environment.

5. License bundling: To reduce the duplication of builds on Sourceforge in which the only difference was the license provided in each, both the Apache and CPL licenses are included in root directory of the Toolkit. Use the one that applies to your situation.
6. Two install options: Two download versions are now offered. The smaller one is for updating existing installations or for reuse in embedded applications that already provide the other processing components--business as usual. A new package with "fullpackage" in the name now incorporates the essential processing modules to create a processing environment for new users and evaluators that requires nothing more than to unzip the file into an appropriate directory and then click on a "start" batch file. A new document in its root directory (an output of doc/EvaluateOT.dita, “Evaluating the DITA Open Toolkit (fullpackage version)”) informs new users how to install and use the Toolkit for the first time.
7. Other enhancements: The public design discussions that fed into the final selection and architectures for this release are documented at the DITA Focus Area in a topic called “DITA OT 1.3 Issues tracking” (http://dita.xml.org/node/1282).

[7 Improvements]

1. Preliminary support for OASIS DITA 1.1
2. Support ICU in index sorting
3. Integrate with Eclipse
4. Refactor Ant script for easy override
5. Topicmerge reimplementation in JAVA
6. Enable XML Catalog Resolver
7. Full package distribution (was GUI/usability)

[21 SF Bugs Fixed]

1. SF Bug 1582506 Docbook cannot handle <author>
2. SF Bug 1548189 Sections should not jump to <h4> for Accessibility reasons
3. SF Bug 1548180 Spaces dropped from index terms
4. SF Bug 1548154 XHTML index links should go to the topic
5. SF Bug 1545038 CommandLineInvoker is unfriendly towards spaces
6. SF Bug 1541055 topicref @id incorrectly uses NMTOKEN type
7. SF Bug 1530443 dost.jar relies on the incorrect behavior of Xerces
8. SF Bug 1473029 Syntax code makes overrides difficult
9. SF Bug 1470101 Metadata in topics is left out of XHTML headers
10. SF Bug 1470077 Choicetable headers create attribute inside attribute
11. SF Bug 1470057 Step template creates attributes after creating tags
12. SF Bug 1465947 <topichead> without children the whole branch to disappear
13. SF Bug 1465941 Keywords defined in map are ignored if <topicref> contains t
14. SF Bug 1465866 Problems in catalog-dita.txt
15. SF Bug 1460447 <morerows> not well supported in pdf tranformation.
16. SF Bug 1457187 'copy-to' doesn't actually copy files
17. SF Bug 1454835 OT renders files referenced via conref only
18. SF Bug 1427808 Should be easier to modify link attributes in XHTML
19. SF Bug 1422182 @colname renaming needs to apply to @namest and @nameend
20. SF Bug 1417820 fo and docbook outputs can\'t handle deep topic dirs
21. SF Bug 1368997 PDF Vertical list of author redundancy

[1 SF Patch Added]

1. SF Patch 1503296 Refactor of HTMLHelp inifiles creation

[1 SF RFE Added]

1. SF RFE 1160960 Enh: Toolkit should work with both both *.dita and *.xml

Note: SourceForge bugs, patches, and RFEs listed above can be found in SourceForge Bugs, Patches, and RFE tracker pages:

- 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 OT release 1.2.2

Release 1.2.2 is a maintenance release to fix defects and make patches based on release 1.2.1.

Improvements

1. Chinese support in WORD RTF
2. Improve plug-in architecture in plug-in dependency handling

SF Changes

1. SF Bug 1461642 Relative paths in toolkit.
2. SF Bug 1463756 TROFF output is not usable
3. SF Bug 1459527 Properties elements should generate default headings
4. SF Bug 1457552 FO gen-toc does not work right for ditamaps and bookmaps
5. SF Bug 1430983 Specialized indexterm does not generate entries in index
6. SF Bug 1363055 Shortdesc disappears when optional body is removed
7. SF Bug 1368403 The dita2docbook transformation lacks support for args.xsl
8. SF Bug 1405184 Note template for XHTML should be easier to override
9. SF Bug 1407646 Map titles are not used in print outputs

10. SF Bug 1409960 No page numbers in PDF toc
11. SF Bug 1459790 Related Links omitted when map references file#topicid
12. SF Bug 1428015 Topicmerge.xsl should leave indentation alone
13. SF Bug 1429400 FO output should allow more external links
14. SF Bug 1405169 Space inside XHTML note title affects CSS presentation
15. SF Bug 1402377 Updated translations for Icelandic
16. SF Bug 1366845 XRefs do not generate page numbers
17. SF Patch 1326450 Make ${basedir} mine
18. SF Patch 1328264 FOP task userconfig file
19. SF Patch 1385636 Tweaks to docbook/topic2db.xsl
20. SF Patch 1435584 Recognize more image extensions
21. SF Patch 1444900 Add template for getting input file URI
22. SF Patch 1460419 Add a new parameter /cssroot:{args.cssroot}
23. SF Patch 1460441 map2hhp [FILES] include
24. SF RFE 1400140 Add a new parameter /cssroot:{args.cssroot}

DITA OT release 1.2.1

Release 1.2.1 is a maintenance release to fix defects and make patches based on release 1.2.

Improvements

1. Corrupt table generated in WORD RTF is fixed
2. Pictures are merged into the WORD RTF instead of creating links to them
3. lq element is supported in WORD RTF
4. Generated text can be translated to different languages in WORD RTF
5. In WORD RTF, if no <choptionhd> given, head will be generated in table

SF Changes

1. SF Bug 1460451 Spaces preserving methods are different among tags.
2. SF Bug 1460449 Nested list can not be well supported.
3. SF Bug 1460445 h2d stylesheet cannot handle HTML files within namespace.
4. SF Bug 1431229 hardcoded path in MessageUtils.java
5. SF Bug 1408477 <desc> element is not handled inside xref for XHTML
6. SF Bug 1398867 ampersands in hrefs (on xref and link) cause build to fail
7. SF Bug 1326439 filtered-out indexterms leak into index through dita.list
8. SF Bug 1408487 Short description is not retrieved for <xref> element
9. SF Bug 1407454 XHTML processing for <alt> is incomplete
10. SF Bug 1405221 Some table frames ignored in dita->xhtml
11. SF Bug 1414398 Cannot set provider for Eclipse help transformation
12. SF RFE 1448712 add support for /plugins directory in plug-in architecture

DITA OT release 1.2

DITA open toolkit Release 1.2 is a major release to add new functions, fulfill new requirements, make some function enhancements and fix bugs over release 1.1.2.1.

Important Change DITA-OT 1.2 offers new error handling and logging system. If you invoke your transformation by using java command line where new error handling and logging system is mandatory, you need to set the *CLASSPATH* Environment Variable on page for `dost.jar`. If you invoke your transformation by using an ant script, you need to do one more step after the setting above. That is adding a parameter in your command to invoke an ant script. For example, use `ant -f ant\sample_xhtml.xml -logger org.dita.dost.log.DITAOTBuildLogger` instead of `ant -f ant\sample_xhtml.xml` to start a transformation defined in ant script file ant\sample_xhtml.xml.

New Functions

1. **New plugin archiecture**

   DITA Open Toolkit 1.2 provides a new function to help users to download, install and use plug-ins and help developers create new plug-ins for DITA Open Toolkit.

2. **Transformation to wordrtf**

   DITA Open Toolkit 1.2 provides DITA to Word transforming function to transform DITA source files to output in Microsoft(R) Word RTF file.

3. **HTML to DITA migration tool**

   DITA Open Toolkit 1.2 provides a HTML to DITA migration tool, which migrates HTML files to DITA files. This migration tool originally comes from the developerWorks publication of Robert D. Anderson's how-to articles with the original h2d code.

4. **Problem determination and log analysis**

   In DITA Open Toolkit 1.2, a new logging method is supported to log messages both on the screen and into the log file. The messages on the screen present user with the status information, warning, error, and fatal error messages. The messages in the log file present user with more detailed information about the transformation process. By analyzing these messages, user can know what cause the problem and how to solve it.

5. **Open DITA User Guide for conditional processing**

   In DITA Open Toolkit 1.2, a new user guide which can help users to use conditional processing is added to toolkit document.

6. **Include the OASIS version langref**

   In DITA Open Toolkit 1.2, a new OASIS version of language reference for DITA standard is added to toolkit document.

7. **Document adapt to OASIS DITA 1.0.1 DTDs**

   DITA DTD files are updated to 1.0.1 version in DITA Open Toolkit 1.2.

Other Changes

1. SF Bug 1304545 Some folders were copied to DITA-OT's root directory
2. SF Bug 1328689 Stylesheet links in HTML emitted with local filesystem paths
3. SF Bug 1333481 Mapref function does not work for maps in another directory
4. SF Bug 1343963 Blank index.html generated for ditamap contains only reltabe
5. SF Bug 1344486 java.io.EOFException thrown out when reading ditaval file
6. SF Bug 1347669 Path Spec. in nested DITA maps
7. SF Bug 1357139 filtering behavior doesn't conform to spec
8. SF Bug 1358619 The property.temp file gets cleaned out by default
9. SF Bug 1366843 XRefs do not generate proper links in FO/PDF
10. SF Bug 1367636 dita2fo-elems.xsl has strange line breaks
11. SF RFE 1296133 Enable related-links in PDF output
12. SF RFE 1326377 Add a /dbg or /debug flag for diagnostic info
13. SF RFE 1331727 Toolkit need to run on JDK 1.5.x(only support to run under Sun JDK 1.5 with saxon in normal case)
14. SF RFE 1357054 Be more friendly towards relative directories
15. SF RFE 1357906 Provide a default output directory
16. SF RFE 1368073 Enable plugins for DITA open toolkit
17. SF RFE 1379518 Clearer error messages and improved exception handling
18. SF RFE 1379523 DITA to Rich Text Format (.rtf) file
19. SF RFE 1382482 plugin architecture of DITA-OT

DITA OT release 1.1.2.1

Release 1.1.2.1 is a full build to provide an urgent fix to fix the following critical problem which users found in release 1.1.2.

- SF Bug 1345600 The build process failed when run "Ant all" in release 1.1.2

For this fix, we have restored all the source DITA files in 'doc' and directories in the binary packages.

Note that the original parameter "args.eclipse.toc" in "Ant tasks and script" was separated to "args.eclipsehelp.toc" for DITA-to-Eclipse help transformation, and "args.eclipsecontent.toc" for DITA-to-dynamic Eclipse content transformation.

Another issue is that we found there is a mismatch in the document and the toolkit behavior when you are trying to use the following command

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

Now we have updated the documentation. Please refer to the topic 'Building DITA output with Java command line' on our website for more details.

These updates do not affect standard operation of the toolkit. The main goal of this minor release to enable new users of the toolkit to run the installation verification tests without failure.

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

Parameter Changes

1. The original parameter "args.eclipse.toc" in "Ant tasks and script" was separated to "args.eclipsehelp.toc" for dita2eclipsehelp transformation, and "args.eclipsecontent.toc" for dita2eclipsecontent transformation.
2. Several parameters were added to the java command line interface, including "/javahelptoc","/javahelpmap","/eclipsehelptoc","/eclipsecontenttoc","/xhtmltoc".

Other Changes

Change to the "doc" directory, except "doc\langref" directory:

1. The source dita files and the generated HTML, CHM, and PDF files were separated into separate downloads.
2. The source package contains the source dita files.
3. The binary package contains the generated HTML, CHM, and PDF files.

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 on page .

   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 on page 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 on page .

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 1.6.5

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 on page 35

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 on page 23 document.

Note: The following instructions and the associated *build.xml* and *ditatargets.xml* files are for the Java 1.4.2, Ant 1.6.5, 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) on page and the XSLT processor on page 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 on page 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 on page 36 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.

Table2. General Parameter Table

Parameter	Description	Required
basedir	The path of the working directory for transformations, it will be the base of relative paths specified by other parameters. **Note:** • If input is relative, it will be set relative to the current directory. • In Ant scripts, the default is that specified in the Ant buildfile. • In Java command line, the default is current directory.	No
dita.dir	The absolute path of the toolkit's home directory.	No
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. ***DEPRECATED*** *- use* `args.input` *instead.*	No
dita.input.dirnam	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. ***DEPRECATED*** *- use* `args.input` *instead.*	No
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	No

Parameter	Description	Required
	dita2docbook.xsl in docbook transformation, dita2fo-shell.xsl in pdf transformation, dita2xhtml.xsl in xhtml/eclipsehelp transformation, dita2rtfImpl.xsl in word transformation and dita2html.xsl in javahelp/htmlhelp transformation.	
dita.input.valfile	The name of the file containing *filter/flagging/revision* information.	No
args.draft	Default "hide draft & required-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

args.logdir	The directory used to keep generated log files. Default will be output directory.
	Note: If several transforms running batchly, e.g., ant all:
	• If the user has specified a common logdir for all transformations, it will be used as log directory.
	• If the user hasn't specified a common dir for all transformations:
	  • If all transformations have same output directory, the common output direcory will be used as log directory.
	  • If there is no same output directory for all transformations, the `basedir` will be used as default log directory.
		No

Table3. 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.outdir}${args.csspath}`. "no" and "yes" are valid values. Default is "no".	No
args.outext	The output file extension name for generated xhtml files. Typically, '.html' or '.htm' can be used as the extension name for the generated xhtml files. You can also specify other extension name. The default is '.html'.	No
args.css	User specified css file, it can be a local file or remote file from website. **Note:** If `${args.csspath}` is an URL, the `${args.css}` should be a filepath relative to the URL.	No
args.cssroot	The root directory of user specified css file. **Note:** If this parameter is set, the `${args.css}` should be a filepath relative to `args.cssroot`.	No
args.csspath	The path for css reference. Default is no path. **Note:** • If `${args.csspath}` is an URL like path, it should starts with `http://` or `https://`. For example: `http://www.ibm.com/css`. • Local absolute paths is not supported for `${args.csspath}`. • Use '`/`' as the path separator and don't append separator at last. For example: `css/mycss`.	No
args.hdf	The name of the file containing XHTML to be placed in the HEAD	No

Parameter	Description	Required
	area.	
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.

Table4. **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
args.eclipse.prov	The provider name of the eclipse help output. The default value is DITA.	No
args.eclipse.vers	The version number of the eclipse help output. The default value is 1.0	No

dita2eclipsecontent
Transform DITA topic or DITA map into Eclipse help plugin for Eclipse dynamic content provider based on xhtml.

Table5. **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
args.eclipse.prov	The provider name of the eclipse help output. The default value is DITA.	No
args.eclipse.vers	The version number of the eclipse help output. The default value is 1.0	No

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

Table6. **Parameter Table of dita2javahelp**

Parameter	Description	Required
args.dita.locale	The locale used for sorting indexterms. If no locale specified, the first occurrence of "xml-lang" will be used as default locale; If no "xml-lang" found, "en-us" will be used by default.	No
args.htmlhelp.inc	The parameter to specify the file that need to be included by the HTMLHelp output.	No

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

Table7. 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
args.dita.locale	The locale used for sorting indexterms. If no locale specified, the first occurrence of "xml-lang" will be used as default locale; If no "xml-lang" found, "en-us" will be used by default.	No

dita2xhtml

Transform DITA topic or DITA map into xhtml web output.

Table8. 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.

Table9. 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
args.fo.output.re	The parameter to specify whether output related links in pdf transformation. "yes" and "no" are valid values. Default is "no". **Note:** Any value that is not "yes" is regarded as "no".	No
args.fo.userconfi	The parameter to specify the user configuration file for FOP.	No

dita2troff

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

dita2wordrtf

Transform DITA topic or DITA map into Word rtf. The `args.artlbl` parameter of the general parameters is not supported.

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.

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.

```
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 this properties to build the sample `sequence.ditamap` file and outputs the xhtml results to the out directory. You can add other parameters to this properties file. See the following *Table of supported parameters on page 40*for details.

Note:

1. In this example, the character slash 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 `${args.logdir}` directory. The following command provides an example using this properties file:

   ```
   ant -f conductor.xml -propertyfile ${args.logdir}/property.temp
   ```

Supported parameters

*Table of supported parameters on page 40*lists the supported parameters (their Ant names are within the braces) that you can set with this tool.

Table10. Table of supported parameters

Parameter	Description
/basedir:{basedir}	The path of the working directory for transformations, it will be the base of relative paths specified by other parameters. **Note:** • If input is relative, it will be set relative to the

Parameter	Description
	current directory. - In Ant scripts, the default is that specified in the Ant buildfile. - In Java command line, the default is current directory.
/ditadir:{dita.dir}	The absolute path of the toolkit's home directory.
/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. ***DEPRECATED*** *- use* `args.input` *instead.*
/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. ***DEPRECATED*** *- use* `args.input` *instead.*
/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, wordrtf and docbook.
/filter:{dita.input.valfile}	The name of the file containing *filter/flagging/revision* information.
/draft:{args.draft}	Default "hide draft & required-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.
/hdf:{args.hdf}	The name of the file containing XHTML to be

Parameter	Description
	placed in the HEAD area.
/csspath:{args.csspath}	The path for css reference. Default is no path. **Note:** - If `${args.csspath}` is an URL like path, it should starts with `http://` or `https://`. For example: `http://www.ibm.com/css`. - Local absolute paths is not supported for `${args.csspath}`. - Use '`/`' as the path separator and don't append separator at last. For example: `css/mycss`.
/css:{args.css}	User specified css file, it can be a local file or remote file from website. **Note:** If `${args.csspath}` is an URL, the `${args.css}` should be a filepath relative to the URL.
/cssroot:{args.cssroot}	The root directory of user specified css file. **Note:** If this parameter is set, the `${args.css}` should be a filepath relative to `args.cssroot`.
/copycss:{args.copycss}	The parameter to specify whether copy user specified css files to the directory specified by `{args.outdir}${args.csspath}`. "no" and "yes" are valid values. Default is "no".
/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, dita2rtfImpl.xsl in word 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.

Parameter	Description
/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.
/provider:{args.eclipse.provider}	The provider name of the eclipse help output. The default value is DITA.
/version:{args.eclipse.version}	The version number of the eclipse help output. The default value is 1.0
/xhtmltoc:{args.xhtml.toc}	The root file name of the output xhtml toc file in xhtml transformation. The default is 'index'.
/logdir:{args.logdir}	The directory used to keep generated log files. Default will be output directory. **Note:** If several transforms running batchly, e.g., ant all: • If the user has specified a common logdir for all transformations, it will be used as log directory. • If the user hasn't specified a common dir for all transformations: • If all transformations have same output directory, the common output direcory will be used as log directory. • If there is no same output directory for all transformations, the `basedir` will be used as default log directory.
/ditalocale:{args.dita.locale}	The locale used for sorting indexterms. If no locale specified, the first occurrence of "xml-lang" will be used as default locale; If no "xml-lang" found, "en-us" will be used by default.
/fooutputrellinks:{args.fo.output.rel.links}	The parameter to specify whether output related links in pdf transformation. "yes" and "no" are valid values. Default is "no". **Note:** Any value that is not "yes" is regarded as "no".
/fouserconfig:{args.fo.userconfig}	The parameter to specify the user configuration file for FOP.
/htmlhelpincludefile:{args.htmlhelp.includefile}	The parameter to specify the file that need to be included by the HTMLHelp output.

Problem determination and log analysis

Introduction

In the DITA Open Toolkit 1.2 or above, a new logging method is supported to log messages both on the screen and into the log file. The messages on the screen present user with the status information, warning, error, and fatal error messages. The messages in the log file present user with more detailed information about the transformation process. By analyzing these messages, user can know what cause the problem and how to solve it.

The logging method is based on Ant's Logger & Listener interface. By default, this logging method is disabled, and all the messages occur on the screen just like previous releases.

To start this new logging method, you need to follow the usage below:

- In Ant command, specify the logger by appendding `-logger org.dita.dost.log.DITAOTBuildLogger` in the command parameters, for example:

  ```
  ant sample.web -logger org.dita.dost.log.DITAOTBuildLogger
  ```

- In Java command, the logger is specified internally, so you do not need to specify it again.

Analyze messages on the screen

During the building process, some information or messages occur on the screen to tell you about the status, warnings, errors, or fatal errors. You need to analyze the messages to solve the problems.

- If the build succeeded with some warning messages on the screen, it means that there are something incorrect within the user input parameters or source DITA files; but you can still get the correct output.
- If the build succeeded with some error messages on the screen, it means that there are something incorrect within the user input parameters or source DITA files; the output maybe not correct.
- If the build failed with fatal error message on the screen, it means that there are something illegal or invalid within the user input parameters or source DITA files; you may get no output, or wrong output.

Analyze messages in the log file

A log file in plain text format is generated in the log directory, which has a name combined with both input file name and transform type. You can open it and find more detailed information, which are helpful for solving problems. You can use the same way introduced above to analyze the messages and solve the problems.

The log directory can be specified by using the parameter `/logdir:{args.logdir}` for the output options.

Note: In some cases, there would be no log file generated:

- You have entered an invalid Ant command or Java command to start the toolkit.
- The log file with the same name in the same directory exists and can not be deleted.

Turn on debug mode

Debug mode is supported along with the new logging method. Under debug mode, diagnostic information, such as: environment variables, stack trace, will be logged into the log file. These information can help the user or developer to go deep into the problems and find the root cause.

By default, the debug mode is disabled. To turn on the debug mode, you need to follow the usage below:

- Append `-d` or `-debug` in Ant command.
- Append `/d` or `/debug` in Java command.

About message file

The message file is used to store the detailed log messages, these messages are read dynamically from this file. To ensure those messages can be read correctly during the transform process, the message file should be located properly. In some situations, the toolkit may fails to load the message file due to some exceptions thrown. Please refer to Troubleshooting on page 49 for detailed information.

For high level users and developers, there is a propery `args.message.file` in the toolkit's ant script, it is used to config the message file, you can overide it in your ant script.

Note: Due to the difference of underly implemetation between Java, And, and XSL, the property `args.message.file` is only useful for Java and Ant; To keep the normal function of log handling, you still need to ensure there are files 'resource/messages.xml' and 'resource/messages.dtd' both in the toolkit's root directory and in the directory that you run the toolkit.

Migrating HTML to DITA

The DITA Open Toolkit release 1.2 or above provides a HTML to DITA migration tool, which migrates HTML files to DITA files. This migration tool originally comes from the developerWorks publication of Robert D. Anderson's how-to articles with the original h2d code. This migration tool is under "demo\h2d" directory. You can use it separately because it is not integrated into the main transformation of toolkit. The version in the toolkit is more recent, but the articles should be referenced for information on details of the program, as well as for information on how to extend it. There are links to the articles at the bottom of this page.

Preconditions

The preconditions to be considered before using the migration tool are listed below:

- The HTML file content must be divided among concepts, tasks, and reference articles. If not, the HTML files should be reworked before migrating.
- This migration tool is intended for topics. The HTML page should contain a single section without any nested sections.
- DITA architecture is focused on topics, information that is written for books needs to be redesigned in order to fit into a topic-based archiecture.
- This migration utility only works with valid XHTML files, HTML files must be cleaned up using HTML Tidy or other utility before processing.

Running examples

You can use the Ant script to migrate only one HTML file or all the HTML files in same directory each time. See Migrating HTML to DITA with Ant script on page for more details.

You can also use the Java command for migration. See Migrating HTML to DITA with Java command on page for more details.

Post conditions

There are also some post conditions to consider after processing:

- In some case, the tool cannot determine the correct way to migrate, it places the contents in a <required-cleanup> element, you should fix such elements in the output DITA files.
- Check the output DITA files. Compare them with the source HTML files and check if both contents are equivalent.

Known limitations

There are some known limitations within the current release, please refer to Known Limitations on page 49 for detailed information.

Extension points

The HTML2DITA migration tool helps extension in the following listed ways:

- The `genidattridbute` template can be overridden to change the method for creating the topic ID.

- The `gentitlealts` template can be overridden to change the ways of title generation.
- Override respond section in the tool to preserve the semantic of source, in case if the <div> or <span> element is used in regular structures.
- You can also migrate to another specialized DTD by overriding the original template base on the specific DTD and your required output.

Additional information

You can find the here original developerWorks publication via links below:

- Migrating HTML to DITA, Part 1: Simple steps to move from HTML to DITA
- Migrating HTML to DITA, Part 2: Extend the migration for more robust results

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">&nbsp;</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!

Known Limitations

Below are some known limitations categoried by module within the current release of the DITA Open Toolkit.

Transformation to PDF and Word RTF

1. You can change the styles of the output file by using tools in Microsoft(R) Word rather than specifying the styles before transforming.
2. If there is a cross reference referring to an URL in the DITA source file, the link should be completed defined with the proper internet protocol. For example, specify `http://www.ibm.com` instead of `www.ibm.com`.
3. Flagging, revision bar and filtering are not supported in PDF and Word RTF output.
4. Morerows attribute of the table element used to generating vertically merged cell is not supported in PDF output.
5. Style attributes for table are not supported in Word RTF output.
6. Complex cases dealing with table in list are not supported in Word RTF.
7. There might be no output style applied on contents of some tags in Word RTF output compared with other output.

HTML to DITA migration

1. Since Xalan doesn't allow to set the public and system IDs dynamically using a variable, when Xalan is used as the default XSLT processor, the output will contain:

   ```
   <!DOCTYPE topic PUBLIC "{$publicid}" "{$systemid}">
   ```

   Suggest to use Saxon as the processor to fix this problem. For other information on this problem, see the section "Other general migration notes" in the first developerWorks article.

2. Currently, the stylesheet can't handle HTML files within namespace like below:

   ```
   <html xmlns="http://www.w3.org/1999/xhtml">
   ```

   Note: This limitation has been fixed in release 1.2.1, please refer to the Migrating HTML to DITA on page 45 for detail information.

Troubleshooting

This section is used for identifying problems when installing and executing the DITA Open Toolkit.

1. Out of Memory Error

In some cases, you might receive a message stating the build has failed due to an "Out of Memory" error. Please follow the steps below to fix this problem:

1. For Windows, type `set ANT_OPTS=%ANT_OPTS% -Xmx256M` in the command prompt, you can also choose to add a new opition `-Xmx256M` to the `ANT_OPTS` environment variable.

For Linux, type `export ANT_OPTS=${ANT_OPTS} -Xmx256M` in the command prompt.

2. Run the transformation again.

2. java.io.IOException: Can't store Document

In some cases, when you run the JavaHelp transformation, you might receive the exception above. This problem is caused by some HTML files unrelated with the current JavaHelp transformation were found under the output directory. Please follow the steps below to fix this problem:

1. Change into the output directory.
2. Clean the output directory.
3. Run the JavaHelp transformation again.

3. Failed to load message file

In some situations, the toolkit may fails to load the message file due to some exceptions thrown.

To fix this problem, you need to check if there are files 'resource/messages.xml' and 'resource/messages.dtd' in the directory that you run the toolkit. If not, please copy them from the toolkit's root directory.

4. Spaces in file names

Spaces in file names will cause trouble during the processing because Ant use space as the delimiter when processing batch files in a list. Please prevent using spaces in the name of dita files.