Ant Quick Start Guide for DITA Open Toolkit

Contents

Preface.....5

Chapter 1: Introduction to the DITA Open Toolkit and Ant.....7

Overview.....8
What is Ant?.....8
What is DITA-OT?.....9
When Should I Use DITA-OT?.....9

Chapter 2: Writing Your Own Build Files.....11

Writing Ant Build Files DITA-OT.....12
Ant Properties for DITA-OT.....13
Generating Documents with DITA-OT.....21
Writing the Build Log to File.....23

Chapter 3: Debugging DITA-OT Transformations.....25

Introducing Document Generation.....26
Understanding PDF Transformations.....27

Best Practices.....29

Glossary.....31

Preface

Explains purpose of this guide, describes the intended audiences, and lists requirements.

This guide provides a quick start for technical writers who are new to both the DITA Open Toolkit and Ant. DITA-OT is first true single-source documentation solution; writers can produce multiple document types from the same XML source files. DITA files are just XML files that conform to the DITA standard. DITA Open Toolkit 1.5 ships with all the DTD and XSD files needed to verify that your DITA-based XML meets the standard. Once you author content that is DITA-compliant, you can include that content in a range of document types.

Ant is a popular build tool used both by software engineers to run software and documentation builds. DITA-OT uses Ant to programatically generate documentation. Most technical writers, and certainly any that write for developer audiences, should learn the basics of Ant. Such an introduction to the tool is beyond the scope of this guide. Rather, the purpose here is to provide just enough working knowledge of Ant to edit and ran DITA-OT builds.

Audience

The primary audience for the guide is technical writers with no experience using the Ant build tool, but have been tasked with evaluating or implementing the use of the DITA Open Toolkit for their organization's documentation. No prior experience with the DITA-OT is assumed.

This audience is assumed, however, to have knowledge of the following technologies:

- XML
- XSL
- DTD
- XSD

The secondary audience is engineers tasked with providing one or more build scripts for a Technical Publications department. They may already be familiar with Ant, but they do not know which Ant properties are relevant to DITA-OT.

Required Software

- DITA-OT 1.5 Full Easy Install, available at http://sourceforge.net/projects/dita-ot/
- A text editor

You do not need to install Ant separately; DITA-OT ships with the required version of Ant.

Guide Overview

The first chapter introduces DITA-OT and Ant and describes when you should consider using the toolkit. The second chapter describes how to edit and write your own Ant build file for documentation projects, provides a reference of Ant properties specific to DITA-OT, and how to generate DITA-based documents from the command line. The third chapter provides tips for debugging transformation errors. A list of recommended practices for writing your build files and a short glossary are also provided.

This guide was written using Notepad++, Gimp2, and DITA-OT 1.5. No proprietary software was used.

Related Documentation

- DITA Open Toolkit User Guide
- DITA 1.1 Language Specification
- Apache Ant 1.7.1 Manual

Chapter 1

Introduction to the DITA Open Toolkit and Ant

Topics:

- *Overview*
- *What is Ant?*
- *What is DITA-OT?*
- *When Should I Use DITA-OT?*

Understanding the role of DITA Open Toolkit and Ant for technical writers.

This chapter further describes the role of DITA Open Toolkit, Ant, and when DITA-OT is a logical choice for your documentation team.

Overview

Introducing the most important authoring tool since FrameMaker.

DITA-based writers gain numerous advantages using this powerful, single-source, document solution. Indeed, I think of DITA as the Holy Grail of technical documentation, the object for which many a manager has sent me on long, fruitless searches in the past, only to return with half-hearted recommendations for a combination of tools that required hours of manual tweaking to reproduce a document in just one alternative format.

Today, I can tell my manager that the Grail has been found, and I can produce a handful of different document output types simultaneously. This is a breakthrough technology for technical writers. In industry jargon, "single source" has previously meant writing in FrameMaker, then importing your source into another expensive application to produce a second output format, typically online help. To gain just a second format from the source often required tedious hours, sometimes days, massaging the text after it had been "translated" into an online help system. DITA-OT allows technical writers to produce seven output formats at the same time.

However, several technologies make this magic happen. Motivated, or just curious, writers will want a more advanced understanding of what makes DITA tick. To do so, you will need to learn how to write Ant build scripts for the DITA-OT and invoke them from the command line.

- *Ant*
- *DITA Open Toolkit*

Hopefully, this guide will motivate you to study DITA-OT further and encourage your publications team to implement a single-source, DITA-based documentation solution.

What is Ant?

Learning about the blurred line between code and documentation and why technical writers need to learn Ant.

If your DITA authoring tool uses the Open DITA Toolkit to generate your documents, you're already using Ant. So, what is Ant, anyway? Ant is a *build tool*, a program used to compile other programs. If you work as a writer in the enterprise software industry, you know that software engineers regularly produce several versions of whatever software they are working on before they release it to the public. Each compilation is called a *build*. Dozens, sometimes hundreds, of builds are compiled before the RTM (release to manufacturing) or GA (general acceptance) build is certified as the official release build. You can often determine the release build of whatever software you are using by reading the Help->About dialog box. For example, my version of XMetal is 5.5.0.219. This means that build 219 was the official release build for XMetal, version 5.5.

Writers also draft, write, revise, and rewrite their documents many times before releasing a document to the public. We tend to call these drafts, rather than builds. You probably saved drafts of your documents in a document repository or CMS, a content management sytem, in the past, but I doubt you thought of your draft, even though it was versioned by the repository, as a software build that either compiled or failed to compile. A successful document "build" meant only that a document opened in your authoring tool the next day, not that all the related documents also opened successfully and "compiled" together to produce a version, albeit incomplete, of the documentation that will eventually make its way to your readers. Hence, the "build" metaphor did not extend beyond the programming code in the engineers' cubicles to the documents crafted by the writers.

DITA changes that forever; the build metaphor is as relevant to you as to the engineers. Behind the user interface of your authoring tool, the DITA Open Toolkit uses Ant to compile a build every time you try to generate your single-source documents. If you want to customize the way DITA-OT generates your documents, you will need to open the hood, so to speak, and get your hands dirty with the internals of the Toolkit and Ant.

What is DITA-OT?

What is the DITA Open Toolkit, anyway?

The DITA Open Toolkit is a popular, free, open-source tool used to transform DITA documents and maps into the output document formats you desire. In fact, most of the proprietary authoring tools use the DITA-OT to transform DITA documentation projects, so you aren't wasting your time learning about how it works. Many errors are more quickly fixed if you understand what the toolkit is doing "beneath the hood" of your authoring tool.

DITA-OT uses Ant to generate your documents. The primary ant script is `build.xml`, which imports several other build scripts to initialize, validate, and transform your .dita documents.

See *Introducing Document Generation* on page 26 for more information about how DITA-OT processes documents.

When Should I Use DITA-OT?

Determining when DITA OT makes sense for your team.

There a several scenarios where using DITA-OT is the appropriate choice for your documentation team, and describing them all is beyond the scope of this guide. However, here are three simple criteria where DITA-OT provides the best solution:

- Your documentation suite contains a lot of content that is reusable for different documents and audiences.
- Your documentation suite contains documentation for developers using the Eclipse IDE.
- Your documentation suite includes both Microsoft HTML Help and PDF-based documents.

DITA-OT is the only publication tool capable of producing both PDF and Eclipse-based documentation from the same source. Eclipse is the industry-standard IDE for many Java developers, and DITA-OT generates the the content and plugin file required for this environment. If your developer audience uses Eclipse, you can easily add IDE-specific online help to your documentation suite.

If your primary audience is end users, rather than software developers and system administrators, you likely need to provide Microsoft HTML Help for them, in addition to PDF-based documentation. DITA-OT is the only tool that produces both from the same source files.

Chapter 2

Writing Your Own Build Files

Topics:

- *Writing Ant Build Files DITA-OT*
- *Ant Properties for DITA-OT*
- *Generating Documents with DITA-OT*
- *Writing the Build Log to File*

Learning about DITA-specific Ant properties.

Generating documents using Ant from the command line. Understanding which Ant properties must be present in an Ant build file for the DITA Open Toolkit.

Writing Ant Build Files DITA-OT

Learning how buildfiles tell Ant what and how to compile.

The sample Ant build scripts provided by the DITA-OT may not be adequate to meet the needs of your documentation project. This topic describes how to customize the default scripts and write your own.

Customize the Default Ant Script

The DITA Open Toolkit contains sample build files for both the DITA-OT and sample documentation. Writers new to the toolkit use the `sample_all.xml` Ant build script to create all the sample documents that come with DITA-OT. The toolkit also contains build scripts for individual ouput types, such as `sample_pdf.xml`. You can modify just one or two Ant properties in these scripts for your own documentation.

Here is the Ant project definition from `template_pdf.xml`

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This file is part of the DITA Open Toolkit project hosted on
     Sourceforge.net. See the accompanying license.txt file for
     applicable licenses.-->
<!-- (c) Copyright IBM Corp. 2004, 2006 All Rights Reserved. -->
<!-- revise @PLACEHOLDER@ names and values-->

<!--
  | basedir can be specified to other places base on your need.
  |
  | Note: input, output, and temp directories will base on the basedir if
  | they are relative paths.
  * -->

<project name="@PROJECT.NAME@_pdf" default="@DELIVERABLE.NAME@2pdf" basedir=".">

  <!-- dita.dir should point to the toolkit's root directory -->
  <property name="dita.dir" value="${basedir}${file.separator}..${file.separator}.."/>

  <!-- if file is a relative file name, the file name will be resolved
       relative to the importing file -->
  <import file="${dita.dir}${file.separator}integrator.xml"/>

  <target name="@DELIVERABLE.NAME@2pdf" depends="integrate">
    <ant antfile="${dita.dir}${file.separator}build.xml" target="init">
      <!-- please refer to the toolkit's document for supported parameters, and
           specify them base on your needs -->
      <property name="args.input" value="@DITA.INPUT@"/>
      <property name="output.dir" value="@OUTPUT.DIR@"/>
      <property name="transtype" value="pdf"/>
    </ant>
  </target>
</project>

```

You simply change the values of the following properties to match the values used in your project:

- Project name: The root element in an Ant build file.
- Target name: Must be one of the DITA-OT targets listed in *Ant Properties for DITA-OT* on page 13

However, the toolkit's scripts assume that your input files are located in same directory structure used by the DITA-OT samples.

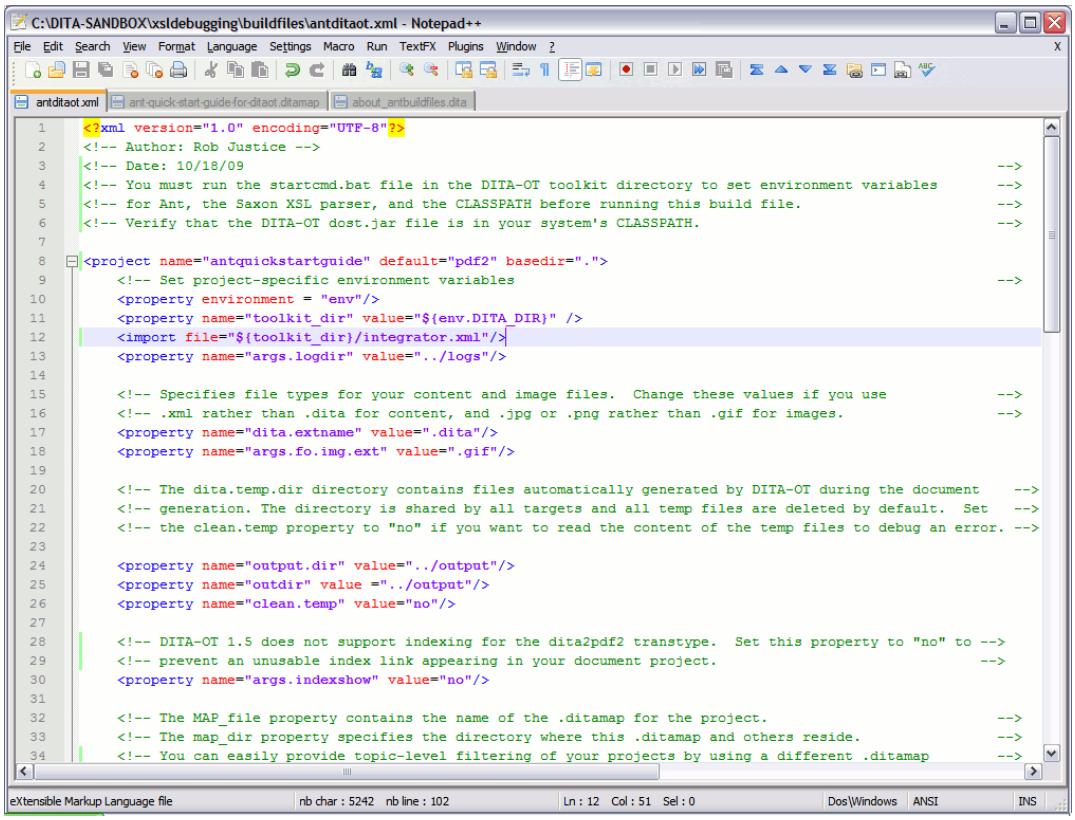
Write Your Own Ant Script

The default build script may not meet the needs of your project for a range of reasons:

- You want to add additional Ant properties not used in the sample template, such XSL and DTD properties to assist your debugging efforts.
- Your content files may not have the same directory structure as the samples.
- You want to place the output files in a different directory.

You need to cutomize or write your own build file for these use cases. For example, each target for this guide's build script uses a separate value for `dita.temp.dir` to assist debugging for a specific output types.

Here is the beginning of the ant script that produced this document:



Ant Properties for DITA-OT on page 13 contains a short list of the most basic Ant properties used by DITA-OT. Use these properties to customize your document's build script for your needs.

Ant Properties for DITA-OT

Reference list of DITA-specific Ant properties.

DITA-OT processes your documentation project as an Ant project, which allows several Ant build properties specific to DITA-OT and your project. These properties can be divided into three categories:

- Properties specific to your documentation project
- Properties specific to the DITA Open Toolkit that you may override
- Properties specific to the DITA Open Toolkit that you should never override

The following tables describes the first group of properties, those specific to your documentation project. The tables also indicate the Java command line option corresponding to the property, if available. Each table describes the properties for one of the following transformation types:

- PDF
- XHTML
- HTMLHelp
- WordRTF
- troff
- docbook
- JavaHelp
- eclipsehelp
- ODT (Open Document format)

Parameters available to all transforms

The following common parameters are available for use by all DITA-OT builds.

Table 1: Common DITA-OT parameters

Project Ant Property	Java Option	Description
`args.debug`	`/debug`	Specifies that DITA-OT print debugging information for your project. Allowed values are "yes" and "no". Default value is "no".
`args.draft`	`/draft`	Indicates whether draft-comment and required-cleanup elements are included in the generated file. Corresponds to XSLT parameter DRAFT in most XSLT modules. Allowed values are "yes" and "no". Default value is "no". If `${publish.required.cleanup}` is not set, `${args.draft}` will be used. **Tip:** The `${publish.required.cleanup}` property is a legacy property that applies only to PDF transformations. The `args.draft` parameter should be used instead. **Tip:** For PDF output, setting `${args.draft}` to "yes" will also cause the contents of `<titlealts>` to appear below the title.
`args.figurelink.style`	NA	Specifies how cross references to figures are styled. Allowed values are "NUMBER" and "TITLE". NUMBER results in "Figure 5", TITLE results in the title of the figure. Corresponds to the XSLT parameter FIGURELINK. **Note:** This parameter is available for all except the PDF transform.
`args.grammar.cache`	`/grammarcache`	Specifies whether to use the grammar caching feature of the XML parser. Allowed values are "yes" and "no". Default value is "yes". **Note:** For most users, this is an important option that dramatically speeds up processing time. However, there is a known problem with using this feature for documents that use XML Entities. If your build fails with parser

Project Ant Property	Java Option	Description
		errors about entity resolution, try setting this parameter to "no".
`args.input`	`/i`	Typically defines the location of the .ditamap file for your documentation project. However, the property can be set to a .dita file, as well. DITA-OT reads this file to find the .dita files that comprise the content for the documentation project.
`args.logdir`	`/logdir`	Defines the location where DITA-OT places log files for your project.
`args.outext`	`/outext`	Specifies the file extension for HTML files in your project's output. Corresponds to XHTML parameter OUTEXT. Default values is ".html".
`args.tablelink.style`	NA	Specifies how cross references to tables are styled. Allowed values are "NUMBER" or "TITLE". The default is "NUMBER", which produces results such as "Table 5". TITLE results in the title of the table. Corresponds to the XSLT parameter TABLELINK.  **Note:** This parameter is available for all except the PDF transform.
`basedir`	`/basedir`	The directory where your project's ant build script resides. The DITA-OT will look for your .dita files relative to this directory. DITA-OT's default build script sets this as an attribute of the project, but you can also define it as a project property.
`dita.ext`	NA	Specifies an extension to use for DITA topics; All DITA topics will use this single extension in the temp directory. Corresponds to XSLT parameter DITAEXT. Default value is ".xml"
`dita.input.valfile`	`/filter`	Defines the location of your project's filter file. Filter files end with the `.ditaval` suffix and are used to filter, include and exclude, content in the generated document. Alternatively, you can create multiple versions of your document by creating a different `.ditamap` file for each version.
`output.dir`	`/outdir`	The location of the directory to hold output from your documentation project.
`transtype`	`/transtype`	Defines the output type for a specific Ant target. Plug-ins may add new values for this option; by default, the following values are available: • PDF • xhtml • htmlhelp • eclipsehelp • eclipsecontent

Project Ant Property	Java Option	Description
		• odt • troff • rtf • javahelp • legacypdf • docbook
`validate`	`/validate`	Specifies whether DITA-OT should validate your content files. Allowed values are "yes" and "no". Default value is "yes".

Parameters available for all XHTML based transforms

The following parameters are available for all output types that are based on the XHTML transform type, including:

- XHTML
- HTMLHelp
- JavaHelp
- eclipsehelp

Table 2: XHTML and related parameters

Project Ant Property	Java Option	Description
`args.artlbl`	`/artlbl`	Adds a label to each image containing the image's filename. Allowed values are "yes" and "no". Default is "no".
`args.breadcrumbs`	NA	Specifies whether to generate breadcrumb links. Corresponds to the XSLT parameter BREADCRUMBS. Allowed values are "yes" and "no". Default is "no".
`args.copycss`	`/copycss`	Indicates whether you want to copy your own .css file to the output directory.
`args.css`	`args.css`	The name of your custom .css file.
`args.csspath`	`/csspath`	The location of your copied .css file relative to the output directory. Corresponds to XSLT parameter CSSPATH.
`args.cssroot`	`/cssroot`	The directory that contains your custom .css file. DITA-OT will copy the file from this location.
`args.ftr`	`/ftr`	Specifies the location of a well-formed XML file containing your custom running-footer for the document body. Corresponds to XSLT parameter FTR.  **Note:** The fragment must be valid XML, with a single root element, common practice is to place all content into .

Project Ant Property	Java Option	Description
`args.gen.default.meta`	NA	Specifies whether to generate extra metadata that targets parental control scanners, meta elements with name="security" and name="Robots". Allowed values are "yes" and "no". Default value is "no". Corresponds to the XSLT parameter genDefMeta.
`args.gen.task.lbl`	`/usetasklabels`	Specifies whether to generate locale-based default headings for sections within task topics. Allowed values are "YES" and "NO". Default is "NO". Corresponds to the XSLT parameter GENERATE-TASK-LABELS. *This parameter is also available for the PDF transform.*
`args.hdf`	`/hdf`	Specifies the location of a well-formed XML file to be placed in the document head.
`args.hdr`	`/hdr`	Specifies the location of a well-formed XML file containing your custom running-header for the document body. Corresponds to XSLT parameter HDR.  **Note:** The fragment must be valid XML, with a single root element, common practice is to place all content into .
`args.hide.parent.link`	NA	Specifies whether to hide links to parent topics in the rendered XHTML. Corresponds to the XSLT parameter NOPARENTLINK. Allowed values are "yes" and "no". Default is "no".
`args.indexshow`	`/indexshow`	Indicates whether `indexterm` element should appear in the output. Allowed values are "yes" and "no". Default is "no".
`args.xhtml.class`	NA	String for a CSS class name attribute applied to the TOC (x)HTML output's <body> element. Found in map2htmltoc.xsl.
`args.xhtml.classattr`	`/xhtmlclass`	Specifies whether to include DITA class ancestry inside generated XHTML elements. Allowed values are "no" and "yes"; the default is "yes" in release 1.5.2 (it was "no" in 1.5 and 1.5.1). For example, the prereq element in a task (which is specialized from section) would generate "`class="section`

Project Ant Property	Java Option	Description
		`prereq`". Corresponds to the XSLT parameter PRESERVE-DITA-CLASS.
`args.xsl`	`/xsl`	Specifies an XSL file that is used rather than the default XSL transform, located in `toolkitdir\xsl \dita2xhtml.xsl`. Property must specify the full path and XSL file name.
`generate.copy.outer`	`/generateouter`	Specifies whether to generate files for content files that are not located in or beneath the directory containing your `ditmap` file.
`onlytopic.in.map`	`/onlytopicinmap`	Specifies whether files that are linked to, or referenced with a conref attribute, should generate output. If set to "yes", only files that are referenced directly from the map will generate output files.
`outer.control`	`/outercontrol`	Specifies whether content files are located in or below the directory containing your `.ditamap` file. The default value is "no." The `gen-list-without-flagging` Ant task generates a harmless warning for content outside the map directory; you can suppress these warnings by setting the `outer.control` property to "true".  Microsoft HTML Help Compiler cannot produce HTMLHelp for documentation projects that use outer content. Your content files must reside in or below the directory containing the `.ditamap` file, and the map file cannot specify ".." at the start of `href` attributes for `topicref` elements.

PDF-specific Ant Properties

The following table describes Ant properties that are specific to the PDF transformation type.

Table 3: PDF parameters

Project Ant Property	Java Option	Description
`args.fo.include.rellinks`	`/ foincluderellinks`	Specifies which links to include in the PDF file. Values are: • "none" (the default) - no links are included. • "all" - all links are included. • "nofamily" - hard coded links and reltable-based links are included. Parent, child, next, and previous links are not included.

Project Ant Property	Java Option	Description
`args.fo.output.rel.links`	`/fooutputrellinks`	 **Note:** This parameter is deprecated in favor of `${args.fo.include.rellin`
		Specifies whether to show links in your project's output. Values are "yes" (include all links) and "no" (the default, include no links). If `${args.fo.include.rellinks}` is specified, this parameter is ignored.
`args.gen.task.lbl`	`/usetasklabels`	Specifies whether to generate locale-based default headings for sections within task topics. Allowed values are "yes" and "no". Default is "no". Corresponds to the XSLT parameter GENERATE-TASK-LABELS. *This parameter is also available for the XHTML based transforms.*
`args.xsl.pdf`	`/xslpdf`	Specifies an XSL file that is used in place of the default XSL transform at `toolkitdir\demo\fo\xsl\fo\topic2fo_shell.xsl`. You must specify the full path and XSL file name.
`retain.topic.fo`	`/retaintopicfo`	Specifies whether to leave the generated FO file for a PDF project.

ODT-specific Ant Properties

The ODT transform, which produces a document using the Open Document Format, is available in the 1.5.2 version of the DITA-OT.

Table 4: ODT related parameters

Project Ant Property	Java Option	Description
`args.odt.img.embed`	`/odtimgembed`	Determines whether images are embedded as binary objects within the ODT file.
`args.odt.include.rellinks`	`/odtincluderellinks`	Specifies which links to include in the ODT file. Values are: • "none" (the default) - no links are included. • "all" - all links are included. • "nofamily" - hard coded links and reltable-based links are included. Parent, child, next, and previous links are not included.

EclipseContent-specific Ant Properties

The "eclipsecontent" transform type produces normalized DITA files, along with Eclipse TOC and project files.

Table 5: EclipseContent properties

Project Ant Property	Java Option	Description
`args.eclipsecontent.toc`	`/eclipsecontenttoc`	Specifies the name of the TOC file for an Eclipse Content project.

XHTML-specific Ant Properties

Parameters in this section are used by the "xhtml" transtype, but not by other XHTML based transforms.

Table 6: Properties for the "xhtml" transform type

Project Ant Property	Java Option	Description
`args.xhtml.toc`	`/xhtmltoc`	Specifies the name of the entry point for an XHTML project. The default value is `index.html`

EclipseHelp-specific Ant Properties

The following table describes Ant properties that are specific to the EclipseHelp transformation type, which is an XHTML based output for use with the Eclipse Help System.

Project Ant Property	Java Option	Description
`args.eclipsehelp.toc`	`/eclipsehelptoc`	Specifies the name of the TOC file.
`args.eclipse.country`	NA	Specifies the more specific region for the language specified with `args.eclipse.language` For example, US, CA and GB would clarify a value of "en" for `args.eclipse.language`. The content will be moved into the appropriate directory structure for an Eclipse fragment.
`args.eclipse.language`	NA	Specifies the base language for translated content, such as "en" for English. This parameter is a prerequisite for `args.eclipse.country`. The content will be moved into the appropriate directory structure for an Eclipse fragment.
`args.eclipse.provider`	`/provider`	Specifies the name of the person or organization providing an Eclipse Help project. Default value is `DITA`. - **DITA Map only**
`args.eclipse.version`	`/version`	Specifies the version number to include in the output. Default value is `0.0.0`. - **DITA Map only**
`args.eclipse.symbolic.nam`	NA	Specifies the symbolic name (aka plugin ID) in the output for an Eclipse Help project. The @id value from the DITA map or the Eclipse map collection (Eclipse help specialization) is the symbolic name for the plugin in Eclipse. By default, the value org.sample.help.doc.- **DITA Map only**

HtmlHelp-specific Ant Properties

The following table describes Ant properties that are specific to the HTML Help compiled help transformation target.

Project Ant Property	Java Option	Description
`args.htmlhelp.includefile/`	`htmlhelpincludefile`	Specifies the name of a file that you want included in an HTMLHelp project.

JavaHelp-specific Ant Properties

The following table describes Ant properties that are specific to the JavaHelp transformation target.

Project Ant Property	Java Option	Description
`args.javahelp.map`	`/javahelpmap`	Specifies the name of the ditamap file for a JavaHelp project.
`args.javahelp.toc`	`/javahelptoc`	Specifies the name of the file containing the TOC in your JavaHelp output. The default value is the name of the ditamap file for your project.

Other Toolkit Ant Properties

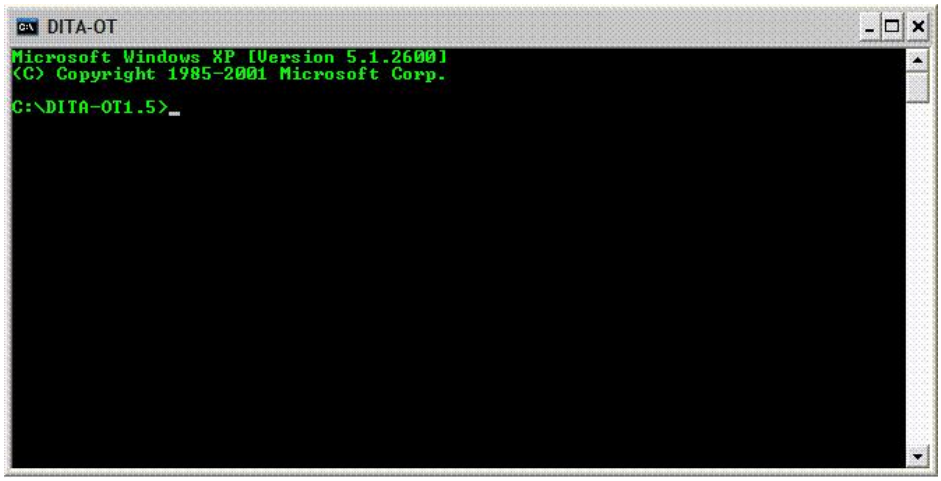
The following table describes additional Ant properties specific to the DITA Open Toolkit that you may override. You should not override a DITA-OT Ant property if it does not appear in this table or one of the tables above.

DITA Ant Property	Java Option	Description
`dita.temp.dir`	`/tempdir`	Defines the directory where DITA-OT will create a temporary directory to place temporary files generated during the transformation process.
`dita.extname`	`/ditaext`	Deprecated. Defines the file extension for content files in the directory specified with the `dita.temp.dir` property. Allowed values are ".xml" and ".dita"; Default is ".xml".
`dita.dir`	`/ditadir`	The location of your DITA-OT installation. Also referred to as `toolkit_dir` in the sample ant scripts. Verify that your project's build script points to the correct location.
`clean.temp`	`/cleantemp`	Specifies whether DITA-OT should delete the files in the temporary directory, `dita.temp.dir`, when it finishes a build. Allowed values are "yes" and "no". Default value is "yes".
`args.dita.locale`	`/ditalocale`	Specifies the language locale file to use for sorting index entries. The JavaHelp transformation type also uses this parameter.

Generating Documents with DITA-OT

How to generate documents from the command line with DITA OT.

1. Open a command prompt.
2. Change directories to where the DITA-OT is installed on your machine.
3. Enter the following command:
   `startcmd.bat` Another command prompt appears with DITA-OT in the title bar, as shown in the following figure:

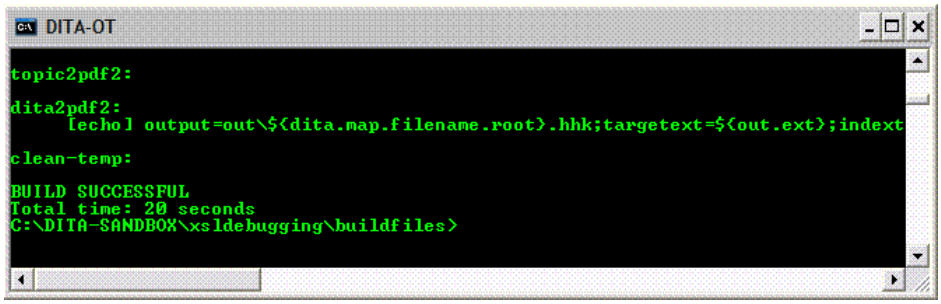


4. Enter the following command and press the Enter key:
   `ant -f buildfile target`

   The following table describes this command.

Syntax	Description
ant	Starts the Ant build tool installed as part of DITA-OT.
-f buildfile	Specifies the build file containing instructions on how to compile your document project. If the build file is not in the current directory, you must specify the path to the file.
target	Specifies the desired output type of the document project. DITA Open Toolkit supports the following targets: • `dita2PDF` • `dita2xhtml` • `dita2htmlhelp` • `dita2eclipsehelp` • `dita2eclipsecontent` • `dita2rtf` • `dita2javahelp` • `dita2troff`

   DITA-OT displays a lot of output in the console window, including whether the build failed or succeeded at the end of the output.



   When your build is unsuccessful, the error message may be difficult to find in the copious output. If you have not configured your console window most of the early output may have already scrolled off the screen. If you add an Ant property, `-logger` to the command line invocation, DITA-OT will save the output to a log file that you can study after the build finishes.

See *Using the DITA-OT Build Logger>* for information about logging DITA-OT build messages to a log file rather than the console window.

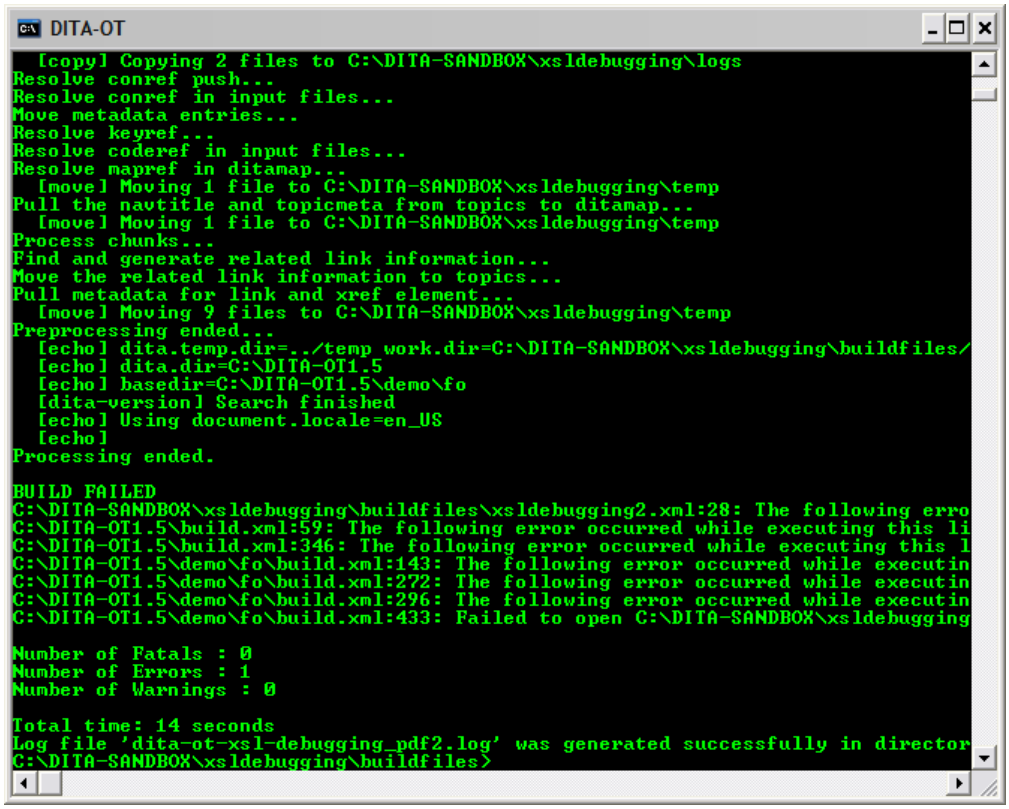
Writing the Build Log to File

How to use the DITA OT build logger

Enter the following command at the command prompt.

```
ant -f buildfile target -logger org.dita.dost.log.DITAOTBuildLogger
```

DITA-OT prints the usual build messages to the console, but also informs you of the name of the log file it generated, as shown in the following figure.



In this example, the log file name is `dita-ot-xsl-debugging_PDF.log`. DITA-OT appends its internal name for the target, `_PDF` to the name of your ditamap to name the log file. If the target had been xhtml, the example log file would be named `dita-ot-xsl-debugging_xhtml.log`. The file contains the same data as the ouput on the console, but you may find it easier to debug xsl transformation errors this way.

Chapter 3

Debugging DITA-OT Transformations

Topics:

- *Introducing Document Generation*
- *Understanding PDF Transformations*

Transforming your DITA-compliant XML into documents.

Understanding the Role of the FO PlugIn. Debugging FO-generated tranformation files.

Introducing Document Generation

Learning the mechanics of document generation with DITA OT.

Your documentation project uses an Ant build script, which calls a target in another Ant build script in the DITA-OT root directory, which imports another Ant build script, which itself imports several more Ant build scripts. Sound confusing? This topic explains this interaction and explains how to identify targets in these scripts related to errors in your document generation.

Each target in the build script for this Quick Start Guide contains the following code snippet.

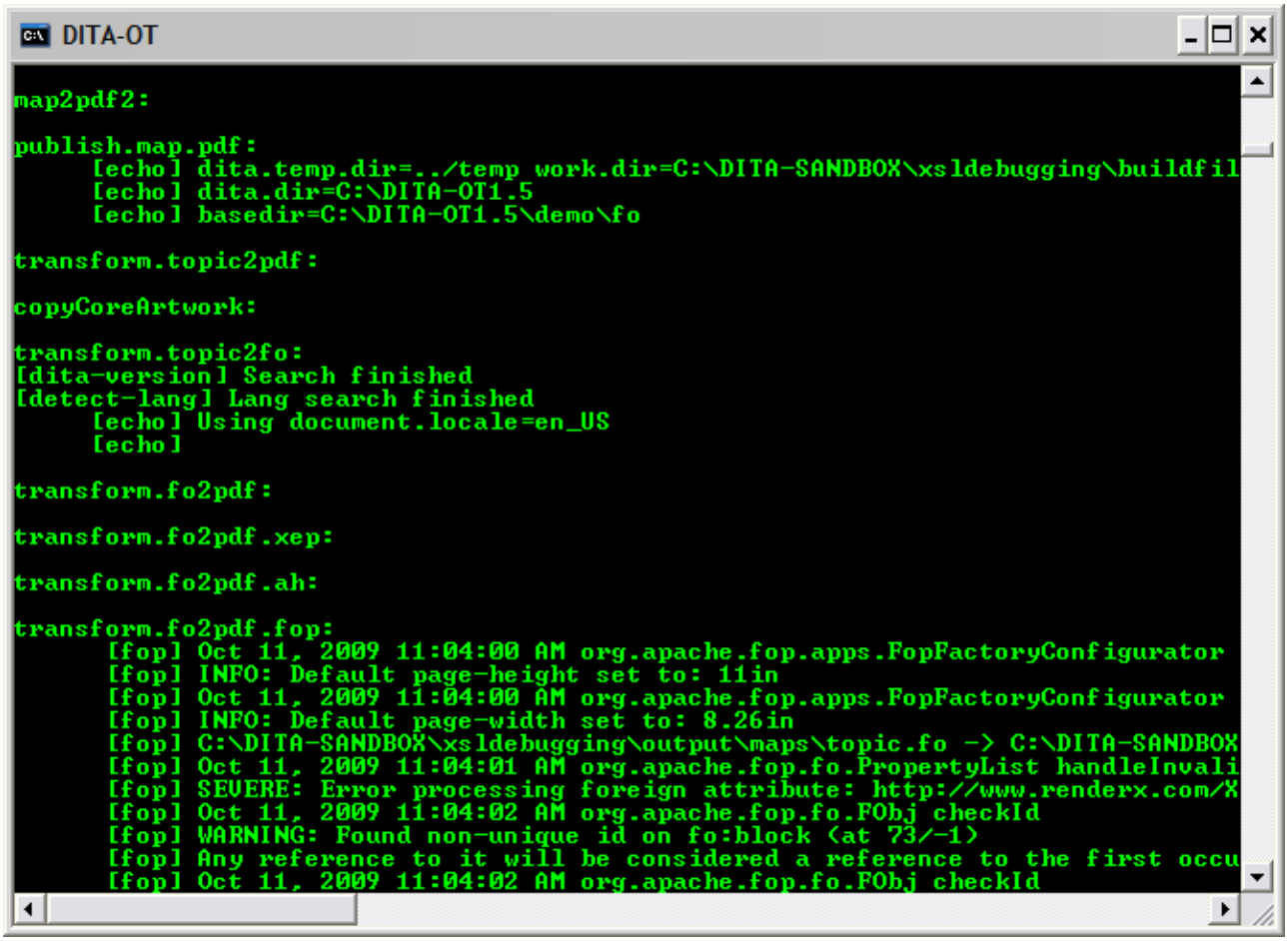
```
<ant antfile="${toolkit_dir}/conductor.xml" target="init">
```

The `toolkit_dir` directory is the root directory where you installed DITA-OT.

The build file you should understand is `build.xml`, located in `toolkit.dir`. The Ant targets defined and imported into this script are the same targets that you see on the console as your build script runs.

DITA-OT Build Script	Description
`build_init.xml`	starts the document transformation, initializes the DITA-OT logger, verifies that the toolkit can locate the files and directories that you specified in your build file, and prints these values to the console and the log file, if you have specified one.
`build_preprocess.xml`	Validates your content files, generates lists of input files, including internal elements distributed across all content, such as index and conref entries. Moves copies of these files and elements into the the directory specified by `output.dir` property in your build script.
`build_general.xml`	Generates XHTML and HTML output from your input files.
`build_dita2wordrtf.xml`, `build_dita2xhtml.xml`, `build_dita2eclipsehelp.xml` `build_dita2javahelp.xml`, `build_dita2htmlhelp.xml`, `build_dita2pdf`	Generate output-specific content from your content files.

Your console displays the name of each Ant target called inside the build scripts, including the output-specific script. For example, the following screen shot displays the names of Ant targets contained in the output-specific `build_dita2pdf.xml` script.



When you see an error in the output, you should read the Ant target that generated it for clues to solve the problem. To learn more about what caused the `INFO`, `SEVERE`, and `WARNING` errors in the image above, you should read the `transform.fo2pdf.fop` Ant target to learn what the Toolkit was doing when the error occurred and which xsl file generated the error.


The DITA-OT build scripts sometimes continue to run even if they are unable to generate a temporary file for one of your content files. The build later displays an error message stating that a DITA-OT build script cannot find a generated file. This error is often misleading; the problem may be that your content file contains an error other than XML validation, which would stop the DITA-OT build from proceeding.

DITA-OT uses a separate set of Ant targets to process your PDF if you specify a value for the `args.fo.userconfig` property in your document's build script. See *Ant Properties for DITA-OT* on page 13 for more information about this property.

Understanding PDF Transformations

Learning how DITA OT uses the FO plugin to generate PDF documents.

DITA-OT performs PDF transformations differently than other output types. A separate software program, known as a renderer or engine, is used to generate the PDF documents. By default, that engine is the FO Plugin, which also uses an Ant build script to generates several files and place them in the directory you specify with the `dita.temp.dir` property in your script. Several of these files are useful for debugging PDF-related transformation errors. However, the FO Plugin build script also deletes them by default. You can override this behavior by setting the value of the `clean.temp` ant property in your build script to "no," as shown in the following code snippet.

```
< property name="clean.temp" value="no"/>
```


The FO Plugin ant build script is located in the following directory: `DITA-OT1.5\demo\fo\build.xml`.

The following table describes the auto-generated files relevant to debugging PDF transformation errors.

File Name	Description
`stage1.xml`	Contains FO processing information related to your index entries.
`stage2.fo`, `stage3.fo`	Contains FO processing information related to your internationalization files.
`buildfile_MERGED.xml`	Contains all the processing information from the generated files and your content in a format that the FO Plugin can use to generate your PDF file.

Examine the content of these files to determine the cause of PDF transformation errors.


The Apache FOP formatter does not support the `indexlist` element for PDF. You can try using a different PDF renderer or engine by specifying a different value for the `pdf.formatter` ant property in the FO plugin build file, or add a `topicref` element in your build file with that links to a manually-created index file.

Best Practices

Tips and tricks for working directly with the DITA OT.

Create targets only for document types that you need.

DITA-OT's most attractive feature is its ability to produce so many different types of documents from the same source files. However, you may find that you need to tweak the targets in your Ant build file to get a document to meet your customization and style guide requirements. Although the sample documents for DITA-OT ship with every available target, there is no point in ironing out the details of a `dita2rtf` target in your build file if your documentation set doesn't require Word-based documents. If you're not providing JavaHelp, troff, or .rtf, then don't create targets for them.

Place all content inside or within the map directory if HTML Help is one of your output types.

The HTML Help Compiler cannot compile the files generated by DITA-OT for source files that reside outside the folder where your `.ditamap` file resides. If your documentation suite contains HTML Help, you should place all your source files in or below this directory.

For advanced debugging, use a different temp folder for each document type within the same build.

The Ant build script for the DITA-OT samples uses a unique folder for each build. However, many builds will include multiple targets, and some of these targets generate overlapping intermediate files. Specify a unique temp directory for each target within the same build to be sure that the intermediate files that you are reading were generated for the target you're debugging. See the build file for this document for an example.

Glossary

build tool
: A program used to compile other programs.

DITA-OT
: DITA Open Toolkit, an implementation of the DITA XML standard.

eclipse
: The Eclipse IDE, integrated development environment, an industry standard tool for Java software developers.

troff
: A processor for the Unix-based groff text processor.