# Installing and upgrading DITA Open Toolkit

[vertical list of authors]
[Anna van Raaphorst]
[Richard Johnson]
[Youyi Zhou]
[Jen Linton]
[JoAnn Hackos]
[Kylene Bruski]


[cover art/text goes here]

# Contents

# Installing and upgrading DITA Open Toolkit

This chapter contains information on how to install and upgrade DITA Open Toolkit on Windows, Linux/UNIX, and Mac OS.

## System requirements and supported applications

### System requirements

DITA Open Toolkit is written in Java and requires at least a minimal set of Java applications be installed. Java SDK 1.5 or 1.6 must be used to execute the applications and the Toolkit Java code.

It is highly likely that any operating system environment where the supported Java SDK can be installed will support basic Toolkit functionality. The Toolkit has been successfully installed and used on Windows XP, Mac OS X, various UNIX and Linux distributions including FreeBSD, Ubuntu Linux, NexentaGNU/OpenSolaris, Solaris, and other operating environments.

Some optional applications can be installed and run only on Windows, for example the HTML Help compiler.

### Required tools

**Note:** All of the required tools except the JDK are bundled in the full package installation. For more information about the installation packages, see Installation considerations on page 6

**Note:** Now the full package ships together with SAXON 9B by default. If you intend to use other XSLT translator, you may need to clean up SAXON 9B libraries first or use other package distributions.

The following tools are required to use DITA Open Toolkit at the 1.4.2 release level.

**Java Development Kits (SDKs)**

Sun. You can download the Sun JDK from http://java.sun.com/javase/downloads/index_jdk5.jsp .

IBM. You can download the IBM JDK from http://www.ibm.com/developerworks/java/jdk .

**Ant**

Ant 1.7.1. You can download Ant from http://ant.apache.org/bindownload.cgi .

**Either the SAXON or Xalan XSLT processor**

SAXON 9B. You can download SAXON from http://saxon.sourceforge.net/ .

Xalan-J 2.7. You can download Xalan from http://archive.apache.org/dist/xml/xalan-j/ .

### Supported languages

DITA and DITA Open Toolkit support the languages listed in the following table.

| Language | xml:lang value |
|---|---|
| Arabic | ar-eg |
| Belarusian | bg-bg |
| Bulgarian | be-by |
| Catalan | ca-es |
| Chinese (Simplified) | zh-cn |
| Chinese (Traditional) | zh-tw |

| | |
|---|---|
| Croatian | hr-hr |
| Czech | cs-cz |
| Danish | da-dk |
| Dutch | nl-nl |
| Dutch (Belgian) | nl-be |
| English (Canadian) | en-ca |
| English (UK) | en-gb |
| English (US) | en-us |
| Estonian | et-ee |
| Finnish | fi-fi |
| French | fr-fr |
| French (Belgian) | fr-be |
| French (Canadian) | fr-ca |
| French (Swiss) | fr-ch |
| German | de-de |
| German (Swiss) | de-ch |
| Greek | el-gr |
| Hebrew | he-il |
| Hungarian | hu-hu |
| Icelandic | is-is |
| Italian | it-it |
| Italian (Swiss) | it-ch |
| Japanese | ja-jp |
| Korean | ko-lr |
| Latvian | lv-lv |
| Lithuanian | lt-lt |
| Macedonian | mk-mk |
| Norwegian | no-no |
| Polish | pl-pl |
| Portuguese | pt-pt |
| Portuguese (Brazilian) | pt-br |
| Romanian | ro-ro |
| Russian | ru-ru |
| Serbian | sr-sp |
| Slovak | sk-sk |
| Slovenian | sl-si |
| Spanish | es-es |
| Swedish | sv-se |
| Thai | th-th |
| Turkish | tr-tr |
| Ukranian | uk-ua |

# Installation overview

Sections in this topic:



## DITA Tookit distributions

DITA Open Toolkit is available in the following distribution formats:

- Full package distributuion
- Small package distribution
- Source distribution

These distributions are all available for download from http://sourceforge.net/projects/dita-ot .

### Full package distribution

The full package distribution contains the Toolkit and most of the basic tools required for doing document builds. Included in the full package are:

- DITA Open Toolkit
- Ant build processor
- XML catalog resolver
- FOP processor for creating PDF outputs
- icu4j (ICU) globalization routines
- SAXON XSLT processor
- Shell scripts for setting the necessary runtime environment variables

To process DITA documents you must also download and install the Java J2SE SDK.

If one of your target output types is HTML Help, you will probably want to install the Microsoft HTML Help compiler. If you one of your target output types is JavaHelp, you will probably want to install the JavaHelp processor.

### Small package distribution

The small package distribution contains only DITA Open Toolkit. You must separately install all the other required and optional processors to create a functioning build environment.

Release 1.2.2 and prior releases of the Toolkit were distributed only in this way.

You might want to download this distribution if you have a prior version of the Toolkit already installed, since the release 1.3 and 1.3.1 Toolkit prerequisites are still the same as those for release 1.2.2.

The small distribution is typically the one used to embed the Toolkit in other products.

### Source distribution

The source distribution contains the source and executable code for the Toolkit (and it also contains the source code for this document). You might download this distribution if you need to modify Toolkit Java code or if you want a detailed look at how the Toolkit works.

## Upgrade considerations

Before upgrading to a new version of DITA Open Toolkit, be sure to back up your current version so you can reapply modifications after your Toolkit upgrade. Such modifications might include:

- Specialization DTDs you added to the `dtd` directory and the corresponding updates you made to the `catalog-dita-template.xml` file
- XSLT stylesheets you have added to the `xsl` directory to override the standard stylesheets
- Plug-ins you have installed
- Catalogs for XML editors

### Installation considerations

Consider the following important points before selecting an authoring tool, and before downloading and installing DITA Open Toolkit and its prerequisite tools.

**The tools you use need to work together as a set.**

This means, for example, that the DITA-aware authoring tool you are already using may not be "aware" of the version of the DTDs that come with the Toolkit. It could also mean that the version of SAXON you already have installed on your laptop doesn't work with the Toolkit. Read System requirements and supported applications on page 3 and check your current system environment before installing or upgrading.

**You may not need to install the prerequisite tools separately.**

DITA Open Toolkit 1.3.1 full package comes with all required tools except the Java JDK.

The Linux distribution Fedora Core 5 already comes with the JDK, Ant, and Xalan. The Mac OS X installation DVD also comes with some of the required components.

**You may need to move or uninstall one or more tools in your current environment before installing the Toolkit and its prerequisites.**

For example, if the version of one of the tools in your Fedora or Mac OS X package is incompatible with the version of the Toolkit you are installing, you may have to change your system environment.

## Installing your DITA authoring tool

You can create DITA source files with a plain text editor, for example Microsoft Notepad for Windows. Editors at varying levels of "DITA-awareness" are also available, both free and for purchase. These editors help you create DITA documents that are well-formed and valid.

For a list of popular DITA-aware authoring tools, see Creating and managing content on page

Authoring tools are not part of DITA Open Toolkit or any of the prerequisite tools for the Toolkit.

Some of these DITA authoring products contain embedded copies of the DITA Open Toolkit and hence you will not need to install the Toolkit separately.

## Installing the Java SDK (JDK)

Both the small and full package distributions require that the Java SDK be installed separately.

### Installing the JDK on Windows

1. For the Sun version of the JDK, enter the URL http://java.sun.com/javase/downloads/index.jsp .
2. From the Sun Developer Network page, scroll to find the heading **JDK 6 Update 3**.
3. Select **Download JDK 6**.
4. From the Sun Developer Network page, accept the license agreement and scroll to the heading "Windows Platform - Java(TM) 2 SDK".
5. Select and download **Windows Installation, Multi-language**.
6. Save and install the `.exe` file.
7. Set the *JAVA_HOME* environment variable to `C:\Program Files\Java\jdk1.6.0_03`.

**Note:** For the IBM version of the JDK, enter http://www.ibm.com/developerworks/java/jdk .

### Installing the JDK on Linux

1. Enter the URL: http://java.sun.com/javase/downloads/index_jdk5.jsp .
2. From the Sun Developer Network page, scroll to find the heading **JDK 5.0 Update 14**.
3. Select **Download**.
4. From the Sun Developer Network page, accept the license agreement and scroll to the heading "Linux Platform - Java Development Kit 5.0 Update 14".
5. Select and download **self-extracting file** .
6. Run and install into a Linux home directory.
7. Set the *JAVA_HOME* environment variable using `export JAVA_HOME=${java_dir}`.

## Installing the DITA Tookit full package distribution

**Installing the full package distribution**

To install the full package:

- Download the full package from http://sourceforge.net/projects/dita-ot .
- Unzip the package into the `C:\ditaot` directory on Windows, or into a home directory on Linux.
- On Windows, create a new shortcut for `C:\ditaot\startcmd.bat` to be used to run DITA builds. On Linux execute the shell script `startcmd.sh` before running a DITA build.

**Note:** You cannot run a DITA Toolkit build until you have installed the Java SDK.
**Note:** You may want to install other optional tools to complete your build environment.

**Verifying the installation**

Unzip or extract the "fullpackage" zip file to a convenient directory, such as your C: drive's root directory. The package will install a directory called `dita-ot1.3.1` that contains not only the usual Toolkit materials but also all the run-time components needed to run the Toolkit in a basic evaluation mode.

Browse to this new directory and double-click on the `startcmd.bat` file in that directory. A new command shell window will open up, with the environment variables already set to enable the Toolkit to run within that shell.

At the command prompt (usually `C:\dita-ot1.3.1` for this version), type `ant samples.web -f build_demo.xml` After a series of processing messages, there should be a new `\out` directory in the `dita-ot1.3.1` directory that contains a folder with the resulting HTML output in it.

Now try the full set of transforms from a single command: `ant all -f build_demo.xml` This command will process every DITA example in the Toolkit into

each of the supported output types. After a much longer flurry of messages stops, the `\out directory` should have a number of folders in it, each with several forms of deliverable produced by the Toolkit demos. If you have the Microsoft HTMLHelp Workshop or the JavaHelp toolset installed, you will even get ready-to use `.chm` and `.javahelp` output files. By comparing the outputs with the various source materials in the distribution, you can get an idea about how the processing works. See Processing (building) and publishing DITA documents on page for more information on processing.

**Demo targets**

```
 all                              Build all output
 clean                            Delete all output
   clean.demo                       Remove the demo output
     clean.demo.book                  Remove the book demo output
     clean.demo.elementref            Remove the Element Reference demo
output
     clean.demo.enote                 Remove the eNote demo output
     clean.demo.faq                   Remove the FAQ demo output
     clean.demo.langref               Remove the Language Reference demo
output
     clean.demo.langref.compilehelp   Remove the Language Reference as
HTML Help output
   clean.doc                        Remove the documentation output
     clean.doc.articles               Delete the articles directory in
doc.
     clean.doc.langref                Delete the langref directory in doc.
   clean.docbook                    Remove the docbook output
   clean.samples                    Remove the sample output
     clean.samples.eclipse            Remove the sample Eclipse output
     clean.samples.htmlhelp           Remove the sample HTMLHelp output
     clean.samples.javahelp           Remove the sample JavaHelp output
     clean.samples.pdf                Remove the sample PDF output
     clean.samples.web                Remove the sample web output
 demo                             Build the demos
   demo.book                        Build the book demo
   demo.elementref                  Build the element reference demo
   demo.enote                       Build the eNote demo
   demo.faq                         Build the FAQ demo
   demo.langref                     Build the Language Reference book as a
demo
   demo.langref.compilehelp         Build the Language Reference as HTML
Help (if the workshop is installed)
 doc                              Build the documentation
   doc.articles.chm                 Build the articles of dita as
document.
   doc.articles.pdf                 Build the articles of dita as
document.
   doc.articles.web                 Build the articles of dita as
document.
   doc.langref.chm                  Build the langref document.
   doc.langref.pdf                  Build the langref document.
   doc.langref.web                  Build the langref document.
 docbook                          Transform the samples to DocBook
 prompt                           Prompt to build anything
 samples                          Build the sample output
   samples.eclipse                  Build the samples for Eclipse
   samples.htmlhelp                 Build the samples for HTMLHelp
   samples.javahelp                 Build the samples for JavaHelp
   samples.pdf                      Build the samples as PDF
   samples.troff                    Build the samples as troff
   samples.web                      Build the samples for the web
```

If you do not specify a target for `build_demo.xml`, the default target is `prompt`.

You can also try your hand at modifying some of the sample scripts in the `ant` directory. These represent the kind of driver files that you would create for your own projects. You can easily adapt these to process your own test DITA files. Run the other ant samples using this example:

```
C:\dita-ot1.3.1>ant -f ant/sample_xhtml.xml
```

This is basically the same as running `ant samples.web -f build_demo.xml`, but intended for you to modify.

You will find the output for this exercise in the `ant` directory itself. You can add parameters to the `sample_xhtml.xml` file to change where your outputs end up, and also to modify the build process in other ways. See Ant processing parameters on page to learn more about processing options.

# Installing the DITA Tookit small package distribution

### Installing the small package distribution

To install the small package:

1. Download the small package from http://sourceforge.net/projects/dita-ot .
2. Unzip the package into the `C:\ditaot` directory on Windows, or into a home directory on Linux.
3. On Windows, add `C:\ditaot\lib;C:\ditaot\lib\dost.jar;C:\ditaot\lib\resolver.jar` to your *CLASSPATH* environment variable.
4. On Linux, set up your environment variable CLASSPATH. For example:

```
export
CLASSPATH=$CLASSPATH:${ditaot_dir}/lib:${ditaot_dir}/lib/dost.jar:${ditaot_dir}/li
```

### Installation considerations

Before installing the DITA Open Toolkit small package and its prerequisite software on Windows, check to see if any of the required tools are already installed on your system and, if so, whether the version you have is supported (see System requirements and supported applications on page 3 ). For any tools you need to install, complete the tasks below in the order shown.

## Installing the small package on Windows

Before installing the DITA Open Toolkit small package and its prerequisite software on Windows, check to see if any of the required tools are already installed on your system and, if so, whether the version you have is supported (see System requirements and supported applications on page 3 ). For any tools you need to install, complete the tasks below in the order shown.

#### Installing Ant on Windows

1. Enter the URL: http://ant.apache.org/bindownload.cgi .
2. On the Apache Ant Project page, find the heading **Current Release of Ant**.
3. Select **apache-ant-1.7.1-bin.zip [PGP] [SHA1] [MD5]**
4. Click **Save** to unzip the apache-ant-1.7.1-bin.zip [PGP] [SHA1] [MD5] file and save it to your C:\ directory as `ant`.
5. Add the `bin` directory to your *PATH* environment variable.
6. Add the *ANT_HOME* environment variable set to `C:\ant`.
7. Add the *ANT_OPTS* environment variable set to `-Xmx256M`.

#### Installing SAXON on Windows

1. Enter the URL: http://saxon.sourceforge.net/ .
2. From SAXON: The XSLT and XQuery Processor page, scroll to find the heading **Saxon-B 9.1**.
3. Select **Download**.

   The SourceForge.net page opens with a list of download options.

4. Select any of the images to start the download.
5. Click **Save** to unzip the `saxonb9-1-0-5j.zip` file and save it to the `C:\` directory as `saxon`.
6. Add `C:\saxon\saxon9.jar;C:\saxon\saxon9-dom.jar;C:\saxon\saxon9-dom4j.jar;C:\s` to the *CLASSPATH* environment variable.
7. Set up ANT_OPTS. For example: `set ANT_OPTS=%ANT_OPTS%`

```
-Djavax.xml.transform.TransformerFactory=
net.sf.saxon.TransformerFactoryImpl
```

**Installing Xalan on Windows**

1. Enter the URL: http://archive.apache.org/dist/xml/xalan-j/ .
2. From Xalan: The Xalan Processor page, scroll to find the heading **xalan-j_2_7_0-bin.zip**. Click to download.
3. Save and unzip the `xalan-j_2_7_0-bin.zip` file to `C:\` directory as `xalan`.
4. Add `C:\xalan\bin` to the *CLASSPATH* environment variable.

**Setting environment variables on Windows**

1. From the Start Menu, select **Start > Settings > Control Panel.**
2. Double-click *System* to open the System Properties window.
3. On the Advanced tab, select *environmental variables* .
4. Modify each *environmental or system variable*.

   Set the *PATH* environment variable to include the directory where you installed the Ant bin directory:
   a. Find the *PATH* environment variable in the list. If *PATH* is not listed, click on **New** under the System variables section.
   b. Type `%ANT_HOME%\bin;%JAVA_HOME%\bin;`
   **> Important:** If there are other variables listed, create a new variable separated by a semicolon. Ensure there are no spaces before or after the semicolon.

   Set the *ANT_HOME* environment variable to the directory where you installed Ant:
   a. Click on **New** under the System variables section.
   b. Type `ANT_HOME` in the variable name field.
   c. Type `C:\ant` in the variable value field.

   Set the *ANT_OPTS* environment variable to the directory where you installed Ant:
   a. Click **New** under the System variables section.
   b. Type `ANT_OPTS` in the variable name field.
   c. Type `-Xmx256M` in the variable value field.

   Set the *JAVA_HOME* environment variable to the directory where you installed the J2SE SDK application:
   a. Click on **New** under the System variables section.
   b. Type `JAVA_HOME` in the variable name field.
   c. Type `C:\j2sdk1.5.0_14` in the variable value field.

   Set the *JHHOME* environment variable to the directory where you installed the JavaHelp application:
   a. Click on **New** under the System variables section.
   b. Type `JHHOME` in the variable name field.
   c. Type `C:\javahelp\jh2.0` in the variable value field.

   Create or append to the *CLASSPATH* environment variable for DITA-OT:
   a. Find the *CLASSPATH* environment variable in the list. If *CLASSPATH* is not listed, click **New** under the System variables section.
   b. Type `C:\ditaot\lib;C:\ditaot\lib\dost.jar;C:\ditaot\lib\resolver.jar`
   **> Important:** If there are other variables listed, create a new variable separated from the others by a semicolon. Ensure there are no spaces before or after the semicolon.

   Create or append to the *CLASSPATH* environment variable for the Apache FOP application:

a. Find the *CLASSPATH* environment variable in the list. If *CLASSPATH* is not listed, click **New** under the System variables section.
b. Type
`C:\fop-0.95\build\fop.jar;C:\fop-0.95\lib\avalon-framework-4.2.0.jar;`
`C:\fop-0.95\lib\batik-all-1.7.jar;C:\fop-0.95\lib\commons-io-1.3.1.jar`
`C:\fop-0.95\lib\commons-logging-1.0.4.jar;C:\fop-0.95\lib\serializer-2`
`C:\fop-0.95\lib\xalan-2.7.0.jar;C:\fop-0.95\lib\xercesImpl-2.7.1.jar;`
`C:\fop-0.95\lib\xml-apis-1.3.04.jar;C:\fop-0.95\lib\xml-apis-ext-1.3.0`
`C:\fop-0.95\lib\xmlgraphics-commons-1.3.1.jar`
**> Important:** If there are other variables listed, create a new variable separated from the others by a semicolon. Ensure there are no spaces before or after the semicolon.

(If you use SAXON) Create or append to environment variables for SAXON:
a. Find the *CLASSPATH* environment variable in the list. If *CLASSPATH* is not listed, click on **New** under the System variables section.
b. Type
`C:\saxon\saxon9.jar;C:\saxon\saxon9-dom.jar;C:\saxon\saxon9-dom4j.jar;`
**> Important:** If there are other variables listed, create a new variable separated by a semicolon. Ensure there are no spaces before or after the semicolon.
c. Set up ANT_OPTS. For example:

```
set ANT_OPTS=%ANT_OPTS%
-Djavax.xml.transform.TransformerFactory=com.icl.saxon.TransformerFactoryImpl
```

(If you use Xalan) Set the *CLASSPATH* environment variable for Xalan:
a. Find the *CLASSPATH* environment variable in the list. If *CLASSPATH* is not listed, click on **New** under the System variables section.
b. Type `C:\xalan\bin`
**> Important:** If there are other variables listed, create a new variable separated by a semicolon. Ensure there are no spaces before or after the semicolon.

**Verifying the installation on Windows**

1. From the toolbar, click `Start > Run`.
2. In the Open field, type `cmd`.
3. Change the command prompt according to the following table.

| If this prompt displays, | type the following command |
|---|---|
| **D:** | **`C:`** |
| **H:** | **`C:`** |
| **C:\My Documents\...** | **`cd \`** |

4. At the prompt, type `cd ditaot`

   The command prompt changes to `C:\ditaot`

5. Type `ant -f build_demo.xml all` and press Enter to process the DITA files in the `demo`, `doc`, `docbook`, and `samples` directories. This procedure also verifies the Toolkit installation.

   The testing process completes in 3-10 minutes depending on the speed of your

machine. When testing completes, the confirmation message BUILD SUCCESSFUL displays.

Be sure the directories and files in your `ditaot` are as described in Directories and files in the ditaot root directory on page 14 .

## Installing the small package on Linux

Before installing DITA Open Toolkit small package on Linux, check to see if any of the required tools are already installed on your system and, if so, whether the version you have is supported (see System requirements and supported applications). on page 3

**Note:** As an example, if you are using Fedora Core 5 Linux, software installation is done using the Package Manager. From this application, if you install the Java Development package within the Development group, you will install:

- Ant 1.7.1
- Java SDK 1.5
- Saxon 9B

For any tools you do need to install, complete the tasks below in the order shown.

**Installing Ant on Linux**

1. Enter the URL: http://ant.apache.org/bindownload.cgi .
2. On the Apache Ant Project page, find the heading **Current Release of Ant**.
3. Select **apache-ant-1.7.1-bin.tar.gz [PGP] [SHA1] [MD5]**.
4. Save and extract the package file into a Linux home directory.
5. Set the *ANT_OPTS* environment variable: `export ANT_OPTS="-Xmx256M"`
6. Set the *ANT_HOME* environment variable to the directory where you installed Ant: `export ANT_HOME=${ant_dir}`
7. Set the *PATH* environment variable to include the directory where you installed the Ant bin directory: `export PATH=${ANT_HOME}/bin:${JAVA_HOME}/bin:${PATH}`

**Installing SAXON on Linux**

1. Enter the URL: http://saxon.sourceforge.net/
2. From SAXON: The XSLT and XQuery Processor page, scroll to find the heading **Saxon-B 9.1**
3. Select **Download for Java**.

   The SourceForge.net page opens with a list of download options.

4. Select any of the images to start the download.

   If SAXON does not appear to be downloading, wait a few minutes before selecting another image. You may have to select more than one image until you find one that works.

5. Download and unzip the `Saxonb9-1-0-5.zip` file and save it to a Linux home directory.
6. Add Saxon to your *CLASSPATH* environment variable: `export CLASSPATH=${CLASSPATH}:${saxon_dir}/saxon9.jar:${saxon_dir}/saxon9-dom.jar`

**Installing Xalan on Linux**

1. Enter the URL: http://archive.apache.org/dist/xml/xalan-j/
2. From SAXON: The Xalan Processor page, scroll to find the heading **xalan-j_2_7_0-bin.tar.gz**. Click to download.
3. Save and unzip the `xalan-j_2_7_0-bin.tar.gz` file to a linux home directory.
4. Add Xalan to the *CLASSPATH* environment variable: `export CLASSPATH=${CLASSPATH}:${xalan_dir}/bin`

**Setting environment variables on Linux**

1. Type in the Linux Console.
2. Modify each *environmental or system variable*.

   Set the *PATH* environment variable to include the directory where you installed the

Ant bin directory: `export PATH=${ANT_HOME}/bin:${JAVA_HOME}/bin:${PATH}`

Set the *ANT_HOME* environment variable to the directory where you installed Ant: `export ANT_HOME=${ant_dir}`

Set the *ANT_OPTS* environment variable to the directory where you installed Ant: `export ANT_OPTS="-Xmx256M"`

Set the *JAVA_HOME* environment variable to the directory where you installed the J2SE SDK application: `export JAVA_HOME=${java_dir}`

Set the *JHHOME* environment variable to the directory where you installed the JavaHelp application: `export JHHOME=${javahelp_dir}`

Set the *CLASSPATH* environment variable for DITA-OT: Set up your environment variable CLASSPATH to include the dost.jar. For example:

```
export CLASSPATH=${ditaot_dir}/lib:${ditaot_dir}/lib/dost.jar:${ditaot_dir}/lib/resolver.
```

Set the *CLASSPATH* environment variable for the Apache FOP application: Set up your environment variable CLASSPATH to include the fop.jar, batik.jar and avalon.jar files in the FOP directory. For example:

```
export CLASSPATH=${fop_dir}/build/fop.jar:${fop_dir}/lib/batik.jar:${fop_dir}/lib/avalon-
```

(If you use SAXON) Set environment variables for SAXON:

a. Set up CLASSPATH to include the saxon.jar file. For example:

```
export CLASSPATH=${CLASSPATH}:${saxon_dir}/saxon9.jar:${saxon_dir}/saxon9-dom.jar:${s
```

b. Set up ANT_OPTS. For example:

```
export ANT_OPTS=${ANT_OPTS} -Djavax.xml.transform.TransformerFactory=com.icl.saxon.TransformerFactoryImpl
```

(If you use Xalan) Set environment variables for Xalan: Set up CLASSPATH to include the `xalan.jar` file and the `xercesImpl.jar` file. For example:

```
export CLASSPATH=${CLASSPATH}:${xalan_dir}/bin
```

**Verifying the installation on Linux**

1. In the console, type `cd {ditaot_dir}`
2. Type `ant -f build.demo.xml all` and press Enter to begin to process the DITA files in the `demo`, `doc`, `docbook`, and `samples` directories. This procedure also verifies the Toolkit installation.

   The testing process completes in 3-10 minutes depending on the speed of your machine. When testing completes, the confirmation message BUILD SUCCESSFUL displays.

Be sure the directories and files in your `ditaot` are as described in Directories and files in the ditaot root directory on page 14 .

## Installing on Mac OS

### MacOS installation considerations

Before installing DITA Open Toolkit and its prerequisite software on Mac OS X, check to see if any of the tools are already installed on your system and, if so, whether the version you have is supported (see System requirements and supported applications). on page 3 ). Java is a core component of Mac OS X. Newer versions of Mac OS X include the full version of the Java JDK 1.4.2 by default. This version of the JDK includes Ant and the Xalan-J XSLT processor, as well. Other tools you want to install may be included on the Mac OS X Developer's Tools on the product DVD.

### Installing on MacOS

To install the Toolkit, extract the zip file to your HOME directory, then edit your login rc file to include the Toolkit in your CLASSPATH.

## Directories and files in the ditaot directory

When you have installed DITA Open Toolkit, the following directories and subdirectories should be in your root `ditaot` directory.

| Directory | Description |
|---|---|
| root (ditaot) | System-level Ant scripts and other system files (for example, `build.xml`, and `integrator.xml`). System-level scripts handle DITA source file processing and transformation into published output. They are an integral part of DITA Open Toolkit and should never be modified by users. For more information, see About Ant scripts on page . |
| css | Sample CSS (cascading style sheet) files. |
| demo | Specializations, plug-ins, and validators that demonstrate extensions to the base DITA language. Includes:<br>• book: bookmap specialization<br>• dita12: DTD and XSD files for DITA 1.2<br>• dita132: DTD and XSD files for DITA 1.3<br>• elementref: simple element reference description markup<br>• enote: data object specialization<br>• faq: faq (frequently asked questions) specialization<br>• fo: plug-in files to produce PDF output<br>• h2d: plug-in to convert XHTML to DITA topics<br>• java: validators for DITA schemas<br>• legacypdf: Legacy PDF transformation plug-in.<br>Many of these directories have README files that provide information about how to use the specializations. |
| doc | DITA documentation: language reference and application notes. |
| dtd | Core DITA definitions in XML DTD format. |
| lib | Contains `dost.jar`, the executable `.jar` file and other |

| | |
|---|---|
| | `.jar` files it depends on. |
| plugins | DITA Open Toolkit plug-ins. |
| resource | Miscellaneous resource files, including the default (common) CSS files and error messages. |
| samples | Sample DITA source files and Ant scripts. |
| schema | Core DITA definitions in XML Schema format. |
| tools | Ant 1.7.1 processor. |
| xsl | Core and process-specific stylesheets. Includes:<br>• common: stylesheets that can be used by any process (for example, internationalization)<br>• docbook: stylesheets used in converting DITA source content into DocBook source<br>• preprocess: code for conditional, conref, and link resolution<br>• troff: stylesheets used in converting DITA source content into troff source<br>• xslfo: code to support the processing of Formatting Objects (FO) output<br>• xslhtml: code to support XHTML processing<br>• xslrtf: code to support RTF processing |

## Installing the optional tools

Sections in this topic:



Depending on the kind of output you expect to produce, you may want to install the following tools.

**(If you plan to publish HTML Help) The Microsoft HTML Help processor**

**(If you plan to publish JavaHelp files) The Sun JavaHelp processor**

**(If you plan to publish PDF files) The Apache FOP processor or the RenderX XEP processor**

The default processing script uses Apache FOP for converting FO files into PDF. With some modification of the build scripts, you can use the RenderX XEP processor, instead. FOP is free. XEP is free for personal use.

**(If you plan to publish Eclipse content) The IBM Eclipse content processor**

For more information, see http://www.eclipse.org/ .

**(If you plan to publish Eclipse help) The IBM Eclipse help processor**

For more information, see http://www.eclipse.org/ .

### Installing the HTML Help compiler on Windows

1. Enter the URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/hwMicrosoftHTMLHelpl
2. From the MSDN page, scroll to find the heading **HTML Help Workshop**.
3. Select **Download Htmlhelp.exe**.
4. Click **Run** and navigate to a C:\ directory as `C:\Program Files\HTML Help Workshop`.
5. Follow the steps in the HTML Help install guide wizard to complete the installation.

If you install the Help compiler to a drive other than the C: drive, you may need to customize the <property> value for `hhc.dir` in some of the build `.xml` scripts in the Toolkit root directory. The Toolkit assumes the compiler is installed on your C: drive.

## Installing FOP on Windows

1. Enter the URL: http://apache.tradebit.com/pub/xml/fop/
2. From the FOP page, in the Name column, select **"fop-0.95-bin.zip"** .
3. Click **Save** to unzip the `fop-0.95-bin.zip` file and save it to the `C:\` directory as `fop-0.95`.
4. Add to the *CLASSPATH* for the following jar files:

```
build/fop.jar
lib/avalon-framework-4.2.0.jar
lib/batik-all-1.7.jar
lib/commons-io-1.3.1.jar
lib/commons-logging-1.0.4.jar
lib/serializer-2.7.0.jar
lib/servlet-2.2.jar
lib/xalan-2.7.0.jar
lib/xercesImpl-2.7.1.jar
lib/xml-apis-ext-1.3.04.jar
lib/xmlgraphics-commons-1.3.1.jar
```

## Installing the JavaHelp processor on Windows

1. Enter the URL: http://java.sun.com/products/javahelp/download_binary.html
2. From the Sun Developer Network page, scroll to find the heading **JavaHelp 2.0_02 (Zip)**.
3. Select **Download**.
4. From the Sun Developer Network page, accept the license agreement and scroll to the heading "Platform - JavaHelp API 2.0_02 FCS"
5. Select **javahelp-2_0_02.zip, 6.49 MB**.

   The `File Download` window opens.

6. Click **Save** to unzip the `javahelp-2_0_02.zip` file and save it to the C:\ directory as `javahelp`.
7. Set the *JHHOME* environment variable to `C:\javahelp\jh2.0`

## Installing FOP on Linux

1. Enter the URL: http://apache.tradebit.com/pub/xml/fop/ .
2. From the FOP page, in the Name column, select **"fop-0.95-bin.tar.gz"** .
3. Save and extract the package file into a Linux home directory.
4. Set the *CLASSPATH* environment variable for the following jar files:

```
build/fop.jar
lib/avalon-framework-4.2.0.jar
lib/batik-all-1.7.jar
lib/commons-io-1.3.1.jar
lib/commons-logging-1.0.4.jar
lib/serializer-2.7.0.jar
lib/servlet-2.2.jar
lib/xalan-2.7.0.jar
```

```
lib/xercesImpl-2.7.1.jar
lib/xml-apis-ext-1.3.04.jar
lib/xmlgraphics-commons-1.3.1.jar
```

```
export
CLASSPATH=${fop_dir}/build/fop.jar:${fop_dir}/lib/avalon-framework-4.2.0.jar:
${fop_dir}/lib/batik-all-1.7.jar:${fop_dir}/lib/commons-io-1.3.1.jar:
${fop_dir}/lib/commons-logging-1.0.4.jar:${fop_dir}/lib/serializer-2.7.0.jar:
${fop_dir}/lib/servlet-2.2.jar:${fop_dir}/lib/xalan-2.7.0.jar:
${fop_dir}/lib/xercesImpl-2.7.1.jar:${fop_dir}/lib/xml-apis-ext-1.3.04.jar:
          ${fop_dir}/lib/xmlgraphics-commons-1.3.1.jar:${CLASSPATH}
```

### Installing the JavaHelp processor on Linux

1. Enter the URL: http://java.sun.com/products/javahelp/download_binary.html
2. From the Sun Developer Network page, scroll to find the heading **JavaHelp 2.0_02 (Zip)**.
3. Select **Download**.
4. From the Sun Developer Network page, accept the license agreement and scroll to the heading "Platform - JavaHelp API 2.0_02 FCS".
5. Select **javahelp-2_0_02.zip, 6.49 MB**.

   The `File Download` window opens.

6. Click **Save** to unzip the javahelp-2_0_02.zip file and save it to a Linux home directory.
7. Add the *JHHOME* environment variable: `export JHHOME=${javahelp_dir}`

## DITAblogs (installing and upgrading)

#### Authoring tools

We recommend using a free, free-trial, or inexpensive DITA-aware authoring tool to create your first demo DITA documents; the experience you gain in a simple environment will help you make the intelligent purchase of a more sophisticated editor later on.

We deliberately chose to use an authoring tool that was free (at the time), since we thought many of our readers would be learning about the DITA Open Toolkit as part of an educational or pilot project where cost might be an issue. We also wanted to edit in "raw" XML ourselves so we would understand that technology well. A couple of months into the project we began exploring more advanced DITA-aware authoring tools that would provide us with additional functionality.

Most of our topics were written using Altova XMLSpy, which is "DITA-aware" (that is, it verifies that DITA source files are well-formed and valid). We had problems at first getting XMLSpy to use a catalog for the DITA DTDs, but that problem was solved (for more information, see Configuring your editor on page  ).

Because our authoring tool didn't have "plausible preview," we did frequent builds to check the output.