
Building Projects with NAnt

Giuseppe Greco <giuseppe.greco@agamura.com>

0.6

Copyright © 2004 Agamura, Inc.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Table of Contents

Introduction	1
Getting NAnt	2
Installing NAnt	2
Running NAnt	3
Project Structure	4
Build Files	6
Targets	9
Configurations	10
A. Build File Examples	11
Master Build File	11
Slave Build File	16
B. GNU Free Documentation License	18
PREAMBLE	18
APPLICABILITY AND DEFINITIONS	18
VERBATIM COPYING	20
COPYING IN QUANTITY	20
MODIFICATIONS	21
COMBINING DOCUMENTS	22
COLLECTIONS OF DOCUMENTS	23
AGGREGATION WITH INDEPENDENT WORKS	23
TRANSLATION	23
TERMINATION	24
FUTURE REVISIONS OF THIS LICENSE	24
ADDENDUM: How to use this License for your documents	24

Introduction

This document provides an introduction to NAnt and describes a set of guidelines for building projects with it.

NAnt is a free build tool based on .NET. NAnt has many advantages over existing build tools that make it the build tool of choice. First of all, NAnt is platform independent. It can be in-

stalled and executed on every system for which a .NET implementation exists. Then, Instead of processing configuration files containing shell-based commands, which are system dependent, NAnt processes build files where targets and tasks are described in XML, making it easier to move projects across systems.

Of course, NAnt does not implement all the functionalities available through shell-based commands, but if you absolutely need a not-implemented functionality, you can either extend NAnt by writing your own task with your preferred .NET programming language, or relay to the general-purpose `<exec>` task, which allows the execution of any program installed on your system.

In order to take full advantage of NAnt capabilities, how projects are structured and how build files are written makes the difference. After introducing how to get, install, and run NAnt, this document presents a set of guidelines that will help you create better projects.

Getting NAnt

NAnt is available at <http://nant.sourceforge.net> and can be downloaded as a zip archive for free.

If you prefer, NAnt can also be checked out from its CVS repository with the following commands:

```
cvs -d :pserver:anonymous@cvs.sourceforge.net:/cvsroot/nant login
```

```
cvs -z3 -d :pserver:anonymous@cvs.sourceforge.net:/cvsroot/nant co nant
```

Then, to maintain your NAnt distribution, up-to-date, run the following command from the project's top directory:

```
cvs -z3 update -dP
```

To learn more about CVS, refer to the official user documentation [<http://www.cvshome.org/docs>].

Installing NAnt

To install NAnt, move to the project's top directory and type

```
nant install
```

Remember to add the `bin` directory to your path.

Running NAnt

Running NAnt is really simple. Just type

```
nant
```

from the project directory containing the build file of interest, and the default target is executed. The default target is specified in the `default` attribute of the `<project>` element. To use a particular target, just specify it after the **nant** command. It is also possible to specify more than one target at once:

```
nant target1 target2
```

When not otherwise specified, NAnt first looks for `default.build`, then for `nant.build`, and at the end, if none of them exist, NAnt processes the first file with the `.build` extension found in the current directory. Use the **-find** option to let NAnt search for build files in parent directories, up to the filesystem root:

```
nant -find target1 target2
```

To make NAnt process a specific build file, use the **-buildfile:<text>** option, where `<text>` specifies the build file to process:

```
nant -buildfile:another.build
```

To see which targets a build file supports, type

```
nant -projecthelp
```

This command also prints target descriptions, if available:

Default Target:

```
build           Builds the current configuration
```

Main Targets:

```
build           Builds the current configuration
clean           Deletes the current configuration
clean-all      Deletes all the configurations
debug          Configures a debug build
dist           Configures a distribution package
init           Initializes building properties
init-package   Initializes packaging properties
install        Installs the current configuration
package        Creates a zip archive ...
release        Configures a release build
src            Configures a source package
test           Tests the current configuration
uninstall      Uninstalls the current configuration
```

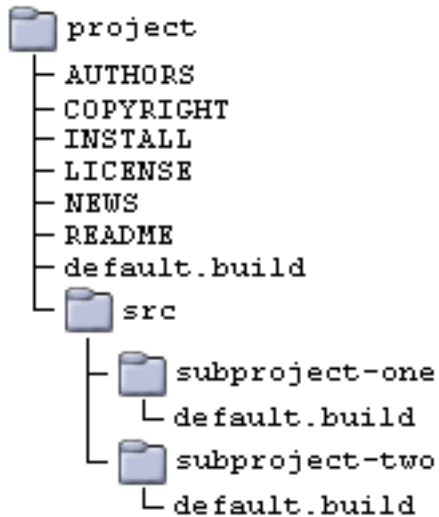
Project Structure

Projects should be well structured. Well structured projects are easier to maintain and contribute to the overall quality.

Large projects should be split up into smaller subprojects in order to keep the number of source files in each directory manageable, and each subproject should be buildable from its own directory as well as from the project's top directory.

Figure 1 shows how projects should be organized, and at least which files should be present in each directory.

Figure 1. Project Structure



Graphical representation of how projects should be organized.

AUTHORS is a text file that contains the name and email address of all the project authors. Each entry in this file should take the following form:

```
Author Name <author.name@author.domain>
```

COPYRIGHT is a text file that contains information about the date(s) and holder(s) of the project copyright. This file should begin with the following two lines:

```
Copyright (C) year Holder
All rights reserved.
```

Optionally, this file may also contain additional information about the holder(s) like address, phone number, email address, etc.

INSTALL is a text file that describes the build and installation procedure.

LICENSE is a text file that contains the licensing conditions. It should clearly state what users are allowed or not allowed to do with binaries, source files, and documentation. For free projects, use the standard licenses [<http://www.gnu.org/licenses>] provided by Free Software Foundation, Inc.

NEWS is a text file that contains a list of worth mentioning changes. For each new release, add one or more items to the top of the file and specify the version they pertain to. Don't discard old items, leave them in the file after the newer ones. This way, users upgrading from any previous version can see what is new.

README is a text file that gives the name of the package and a general description of what it does. This file should also refer to where build and installation instructions can be found.

Projects should at least contain the `src` directory, which contains the project source files. As mentioned above, large projects should be split up into smaller subprojects, and in that case, `src` should contain a subdirectory for each subproject.

Additional executables necessary to build the project should be located in the `bin` directory, while user documentation source files should be placed in the `doc` directory.

Each project directory should contain a `.build` file, and it should be possible to start a build from any level of the project hierarchy.

Build Files

A *Build file* is an XML file used to tell NAnt how to build a project. To see how NAnt processes build files, let's take a look at a very simple scenario -- building a small console application written in C#. Here is the program code:

```
public class HelloWorld
{
    static void Main()
    {
        System.Console.WriteLine("Hello World!");
    }
}
```

To produce the executable binary *helloworld.exe*, you can either use the C# command line compiler, or write a build file and let NAnt build it for you:

```
<?xml version="1.0"?>
<project
  name="helloworld"
  default="build">
  <target name="build">
    <csc
      target="exe"
      output="helloworld.exe">
      <sources>
        <include name="*.cs" />
      </sources>
    </csc>
  </target>
</project>
```

NAnt will then parse the content of the build file and perform the specified tasks.

Of course, using NAnt for projects comprised of a single class like this is massive overkill. But what if you were interested in first building the executable, and then running it? Or what if you want to build one or more dependencies and then build the main executable? This is where NAnt can save loads of time.

A build file is primarily comprised of *targets*, *tasks*, and *dependencies*. A *tasks* is a single action you want NAnt to perform. Examples of tasks include running a compiler, copying or deleting files, sending emails, zipping up sets of files, etc. For a complete list of the tasks supported by NAnt, refer to the official user documentation [<http://nant.sourceforge.net/help/index.html>].

A *target* represents a set of tasks you want NAnt to perform. Targets enable you to group tasks together logically. So if you want NAnt to delete the contents of the `bin` directory, compile five executables, and copy the result somewhere, these actions could be grouped together into a single target.

A *dependency* can be seen as a relationship between targets. For example, in the build file above there is a single target. Its name is `build` and it runs the compiler against a single source file. Setting the `default` attribute of the `<project>` tag to `build` causes the `build` target to be processed by default.

Now let's add a second target, one that will execute *helloworld.exe* after it is compiled:

```
<?xml version="1.0"?>

<project
  name="helloworld"
  default="run">

  <target
    name="build"
    description="Builds HelloWorld">
    <csc
      target="exe"
      output="helloworld.exe">
      <sources>
        <include name="*.cs" />
      </sources>
    </csc>
  </target>

  <target
    name="run"
    depends="build"
    description="Runs HelloWorld">
    <exec program="helloworld.exe" />
  </target>

</project>
```

The new `run` target contains a single action, `exec`, which executes the program. It also con-

tains a dependency on the `build` target. This means that before the `run` target will be executed, the `build` target must also be executed, and it must complete successfully. Note that the default attribute of the `<project>` tag has been changed to `run`. Because `run` depends on `build`, this ensures that the application will be built before it is run. If the `build` target fails, the `run` target won't be executed, and the compiler's error message is displayed to the console.

NAnt won't run the compiler if you have a compiled binary that's newer than the source code. In addition, if multiple dependencies exist in a build file, NAnt is smart enough to only build the dependent component once. But in some cases, you may want to wipe out all existing compiled binaries and perform a clean build:

```
<?xml version="1.0"?>

<project
  name="helloworld"
  default="run">

  <target
    name="build"
    description="Builds HelloWorld">
    <mkdir dir="bin" />
    <csc
      target="exe"
      output="bin/helloworld.exe">
      <sources>
        <include name="*.cs" />
      </sources>
    </csc>
  </target>

  <target
    name="clean"
    description="Deletes compiled binaries">
    <delete
      dir="bin"
      failonerror="false" />
  </target>

  <target
    name="run"
    depends="build"
    description="Runs HelloWorld">
    <exec program="bin/helloworld.exe" />
  </target>

</project>
```

You probably don't want to run the `clean` target every time you build, but just every once in a while. To execute the `clean` target, type

nant clean

This causes NAnt to execute *only* the `clean` target. You can also see that in this version of the build file we have added a `mkdir` task to create a separate `bin` subdirectory, into which compiled binaries are deposited. To clean its contents and rebuild your project, type

```
nant clean build
```

Targets

According to the project structure shown in Figure 1, each project directory should contain a build file that at least supports the standard targets defined for the type to which it belongs. There are mainly two build file types:

1. *Master* build file
2. *Slave* build file

A *master* build file provides targets for creating zip archives, installing artifacts, generating documentation, and of course, building the project itself. But most of the times, building a project means building a number of subprojects, which are not built directly by the master build file. Instead, the master build file just delegates to one or more *slave* build files. Normally, there is only one master build file per project, and it resides in the project's top directory. Master build files should at least support the targets described in Table 1.

Table 1. Target Supported by Master Build Files

Target	Description
<code>build</code>	Builds the project. This target runs NAnt on all subprojects' build files by specifying <code>build</code> as the target. Most of the times <code>build</code> is the default target.
<code>test</code>	Performs unit tests. This target runs NAnt on all subprojects' build files by specifying <code>test</code> as the target.
<code>clean</code>	Deletes all the files generated by the <code>build</code> , <code>test</code> , and <code>doc</code> targets. After executing this target, the project is in a clean slate.
<code>install</code>	Installs project artifacts on the native system. This target depends on the <code>build</code> and <code>doc</code> targets.
<code>uninstall</code>	Uninstalls project artifacts from the native system.
<code>package</code>	Creates a zip archive containing the distribution files (executables, documentation, etc.). To create a zip archive containing the source files only, just specify the <code>src</code> configuration before the <code>package</code> target. More on configurations in the section called "Configurations".

A *slave* build file provides the targets for building and testing a subproject. Slave build files should function either when processed directly by invoking NAnt from the location where they reside, or when called through the master build file or any other slave build file. Slave

build files should at least support the targets described in Table 2.

Table 2. Target Supported by Slave Build Files

Target	Description
build	Builds the current subproject. Most of the times this is the default target.
test	Performs unit tests. Tests should be based on NUnit [http://www.nunit.org].
clean	Deletes all the files generated by the build and test targets. After executing this target, the subproject is in a clean slate.

Appendix A, *Build File Examples* provides an example for each build file type discussed in this section.

Configurations

Configurations let you modify target behaviors. A configuration is just a target that sets a number of properties associated to a another target.

Classical example of configurations are *debug* and *release*. As their names suggest, they are used to configure a *debug* or a *release* build, respectively. Let's see a build file fragment that illustrates how to implement configurations:

```
<?xml version="1.0"?>
<project
  name="helloworld"
  default="run">
  ...
  <target
    name="debug"
    description="Configures a debug build">
    <property
      name="project.config"
      value="debug" />
    <property
      name="build.debug"
      value="true" />
    <property
      name="package.name"
      value="{nant.project.name}-{project.config}" />
  </target>

  <target
    name="release"
    description="Configures a release build">
    <property
      name="project.config"
```

```
    value="release" />
  <property
    name="build.debug"
    value="false" />
  <property
    name="package.name"
    value="{nant.project.name}" />
</target>

<target
  name="build"
  description="Builds HelloWorld">
  <mkdir dir="{build.dir}/{package.name}/bin" />
  <csc
    target="exe"
    debug="{build.debug}"
    output="{build.dir}/{package.name}/bin/helloworld.exe">
    <sources>
      <include name="*.cs" />
    </sources>
  </csc>
</target>

...
</project>
```

Depending on which configuration is executed, the `build` target will behave differently. For example, the command

```
nant debug build
```

tells NAnt to first execute the `debug` target, which is actually the configuration, and then to execute the `build` target. The `debug` configuration sets the `build.debug` property to `true`, causing the `build` target to generate a *debug* version of the executable.

Appendix A, *Build File Examples* provides a complete example of how to implement configurations.

A. Build File Examples

This appendix presents some examples of how to write *master* and *slave* build files. A complete example project can be downloaded from Agamura's developer site [<http://developer.agamura.com/resources/>].

Master Build File

```
<?xml version="1.0"?>

<project
  name="helloworld"
  default="build">

  <sysinfo />

  <!-- global framework settings -->
  <property
    name="target.framework"
    value="\${framework::get-target-framework}" />
  <property
    name="assembly.dir"
    value="\${framework::get-assembly-directory(target.framework)}" />

  <!-- global project settings -->
  <property
    name="project.version"
    value="1.0" />
  <property
    name="build.dir"
    value="build" />
  <property
    name="install.dir"
    value="/usr/local"
    if="\${platform::is-unix}" />
  <property
    name="install.dir"
    value="\${sys.env.ProgramFiles}"
    if="\${platform::is-win32}" />

  <!-- default configuration -->
  <property
    name="project.config"
    value="debug" /> <!-- debug|release -->
  <property
    name="package.config"
    value="dist" /> <!-- dist|src -->

  <!-- named configurations -->
  <target
    name="init"
    description="Initializes building properties">
    <call target="\${project.config}" />
  </target>

  <target
    name="init-package"
    description="Initializes packaging properties">
    <call target="\${package.config}" />
  </target>

  <target
    name="debug"
    description="Configures a debug build">
    <property
```

```
    name="build.debug"
    value="true" />
  <property
    name="package.name"
    value="${nant.project.name}-${project.version}-${project.config}" />
</target>

<target
  name="release"
  description="Configures a release build">
  <property
    name="project.config"
    value="release" />
  <property
    name="build.debug"
    value="false" />
  <property
    name="package.name"
    value="${nant.project.name}-${project.version}" />
</target>

<target
  name="dist"
  description="Configures a distribution package">
  <property
    name="dist"
    value="true" />
  <property
    name="src"
    value="false" />
  <property
    name="archive.name"
    value="${package.name}.zip" />
  <fileset
    id="archive.set"
    basedir="${build.dir}">
    <include name="../AUTHORS" />
    <include name="../COPYRIGHT" />
    <include name="../LICENSE" />
    <include name="${package.name}/**" />
  </fileset>
</target>

<target
  name="src"
  description="Configures a source package">
  <property
    name="package.config"
    value="src" />
  <property
    name="dist"
    value="false" />
  <property
    name="src"
    value="true" />
  <property
    name="temp.dir"
```

```

    value="/tmp"
    if="${platform::is-unix()}" />
<property
  name="temp.dir"
  value="${sys.env.TEMP}"
  if="${platform::is-win32()}" />
<property
  name="archive.src.dir"
  value="${temp.dir}/${nant.project.name}-${project.version}" />
<property
  name="archive.name"
  value="${nant.project.name}-${project.version}.src.zip" />
<fileset
  id="archive.set"
  basedir="${temp.dir}">
  <include name="${nant.project.name}-${project.version}/**" />
</fileset>
</target>

<!-- build tasks -->
<target
  name="build"
  depends="init"
  description="Builds the binaries for the current configuration">
  <nant target="${project.config} build">
    <buildfiles basedir="src">
      <include name="**/*.build" />
    </buildfiles>
  </nant>
</target>

<target
  name="test"
  depends="init"
  description="Tests the current configuration">
  <nant
    target="${project.config} ${target::get-current-target()}">
    <buildfiles basedir="src">
      <include name="**/*.build" />
    </buildfiles>
  </nant>
</target>

<target
  name="clean"
  depends="init"
  description="Deletes the current configuration">
  <delete
    dir="${build.dir}/${package.name}"
    failonerror="false" />
</target>

<target
  name="clean-all"
  description="Deletes all the configurations">
  <delete
    dir="${build.dir}"

```

```

        failonerror="false" />
    <delete failonerror="false">
        <fileset basedir=".">
            <include name="*.zip" />
        </fileset>
    </delete>
</target>

<target
    name="install"
    depends="build"
    description="Installs the current configuration">
    <mkdir
        dir="${install.dir}/${package.name}"
        failonerror="false" />
    <copy
        todir="${install.dir}/${package.name}"
        overwrite="true">
        <fileset basedir="${build.dir}/${package.name}">
            <include name="doc/**" />
            <include name="bin/*.dll" />
            <include name="bin/*.exe" />
        </fileset>
    </copy>
</target>

<target
    name="uninstall"
    depends="init"
    description="Uninstalls the current configuration">
    <delete
        dir="${install.dir}/${package.name}"
        failonerror="false" />
</target>

<target
    name="package"
    depends="init init-package"
    description="Creates a zip archive for the current configuration">
    <delete
        file="${archive.name}"
        failonerror="false" />
    <if test="${src}">
        <mkdir
            dir="${archive.src.dir}"
            failonerror="false" />
        <copy
            todir="${archive.src.dir}"
            overwrite="true">
            <fileset basedir=".">
                <include name="**" />
                <exclude name="*.zip" />
                <exclude name="**/build/**" />
            </fileset>
        </copy>
    </if>
    <if test="${dist}">

```

```
    <call target="build" />
  </if>
  <zip zipfile="{archive.name}">
    <fileset refid="archive.set" />
  </zip>
  <if test="{src}">
    <delete
      dir="{archive.src.dir}"
      failonerror="false" />
  </if>
  <echo
    message=
      "Created archive at file://{nant.project.basedir}/{archive.name}"
  </target>
</project>
```

Slave Build File

```
<?xml version="1.0"?>

<project
  name="helloworld"
  default="build">

  <!-- global framework settings -->
  <property
    name="target.framework"
    value="{framework::get-target-framework}" />
  <property
    name="assembly.dir"
    value="{framework::get-assembly-directory(target.framework)}" />

  <!-- global project settings -->
  <xmlpeek
    file="../../HelloWorld.build"
    xpath="/project/property[@name = 'project.version']/@value"
    property="project.version" />
  <property
    name="assembly"
    value="HelloWorld"/>
  <property
    name="build.dir"
    value="../../build" />

  <!-- default configuration -->
  <property
    name="project.config"
    value="debug" /> <!-- debug|release -->

  <!-- named configurations -->
  <target
```

```
    name="init"
    description="Initializes build properties">
    <call target="{project.config}" />
</target>

<target
  name="debug"
  description="configures a debug build">
  <property
    name="build.debug"
    value="true" />
  <property
    name="package.name"
    value="{nant.project.name}-{project.version}-{project.config}" />
</target>

<target
  name="release"
  description="configures a release build">
  <property
    name="project.config"
    value="release" />
  <property
    name="build.debug"
    value="false" />
  <property
    name="package.name"
    value="{nant.project.name}-{project.version}" />
</target>

<!-- build tasks -->
<target
  name="build"
  depends="init"
  description="Builds the binaries for the current configuration">
  <echo message="Build Directory is {build.dir}/{package.name}" />
  <mkdir
    dir="{build.dir}/{package.name}/bin"
    failonerror="false" />
  <csc
    target="exe"
    debug="{build.debug}"
    output="{build.dir}/{package.name}/bin/{assembly}.exe">
    <sources failonempty="true">
      <include name="*.cs" />
    </sources>
  </csc>
</target>

<target
  name="clean"
  depends="init"
  description="Deletes the current configuration">
  <delete failonerror="false">
    <fileset basedir="{build.dir}/{package.name}/bin">
      <include name="{assembly}.exe" />
      <include name="{assembly}.pdb" />
    </fileset>
  </delete>
</target>
```

```
    </fileset>
  </delete>
</target>

<target
  name="*"
  description="Handles unknown targets">
  <echo message="skip" />
</target>
</project>
```

B. GNU Free Documentation License

FSF Copyright note

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the

license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another lan-

guage. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before re-distributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

GNU FDL Modification Conditions

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" sec-

tion. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical In-

variant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer,

the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Samle Invariant Sections list

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

Sample Invariant Sections list

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.