



User Guide

JWare/AntXtras Log4Ant

Release 2.0.0



Sandbox Software MC.
1133 Broadway, Suite 706
New York, NY. 10010
<http://purl.net/jware/>

Simone Cato
Version 2.0.0, Revised Jan 2010

This document is copyright 2002-2010, Sandbox Software MC (SSMC).

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 published by the Free Software Foundation; with the Front-Cover Texts being “User Guide JWare/AntXtras Log4Ant”. A copy of the license is included in the section entitled “GNU Free Documentation License”.

The following are trademarks of other organizations:

- **Java, J2EE, JEE, J2SE, Sun, Sun Microsystems and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.**
- **UNIX is a registered trademark of The Open Group.**
- **Linux is freely distributed under the GNU General Public License (GPL). The term "Linux" is a registered trademark of Linus Torvalds, the original author of the Linux kernel.**
- **Windows, Microsoft, and Win32 are registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.**
- **All other trademarks cited in this document are the property of their respective owners.**

Table of Contents

Introduction	5
Terms Of Use.....	5
Log4Ant Quick Start Review.....	5
How To Read Descriptions	5
Installation.....	6
Log4Ant Components	7
LibCheckTask <emit:libcheck>.....	8
EmitIncludeItemSet <emit:includes>.....	9
EmitConfigurationType <emit:configuration>	11
EmitManageTask <emit:manage>.....	14
EmitOverlayTaskSet <emit:overlay>	16
EmitCaptureLogsTaskSet <emit:capturelogs>.....	19
EmitMappings <emit:mappings>	21
EmitTask <emit:show>	23
CheckpointTask <emit:checkpoint>.....	26
Log4Ant Antlib Definitions	28
GNU Free Documentation License.....	29

Introduction

This document is the official user reference of the JWare/AntXtras Logging for Ant package (Log4Ant); it gives you detailed descriptions, including examples, of the public Log4Ant *script* components. This document does not contain any extended Log4Ant tutorials; nor does it describe how you can extend Log4Ant to create your own components. For this type of information you can visit the JWare Software website (jwaresoftware.org/wiki/log4ant/).

We assume you have already installed a Log4Ant distribution and all its dependencies, and have read the general Log4Ant Overview on the JWare Software website.

Terms Of Use

The JWare/AntXtras Logging for Ant package (Log4Ant), binary and source form, is released under the Free Software Foundation's GNU LGPL; a copy of [this license](#) can be found on the Free Software Foundation's website. *Please read the LGPL carefully before using any of the JWare/AntXtras source in your own application.*

Log4Ant uses software developed by and on behalf of the Apache Software Foundation, <http://www.apache.org/>.

This JWare Series documentation is copyright Sandbox Software MC. All rights reserved. Read the "GNU Free Documentation License" section at the end of this document for information.

Log4Ant Quick Start Review

The Log4Ant components provide a full bridge to the SLF4J library. SLF4J is a simple façade for various logging frameworks like Log4J, Jakarta Commons Logging, LOGBack, and the JRE's own `java.util.logging` module. By using Log4Ant components you can easily capture, transform, and send Ant messages, AntXtras messages, and standard output and error messages to your real logging and monitoring system. *This means you can capture detailed information about your build, test, and deploy processes for live monitoring or subsequent post-run inspection.*

Log4Ant has three main objectives: first, it wants to get *all* Ant sourced messages into your actual logging or monitoring system; second, it wants to provide a rich set of feedback components for you to use instead of the basic `<echo>` task that Ant provides; and third, it wants to accomplish the previous two objectives without forcing you to rewrite your existing scripts or to do a lot of post-capture massaging to get the outputs to show only what you need.

How To Read Descriptions

Each Log4Ant component description contains the component's default antlib name, its category, a brief description, the list of parameters, the list of nestable elements (if any), a set of examples, and a list of related items (if any). Although each item's description is labeled using our conventions, the full Java class name is also supplied so you can redefine the component however you want in your own antlib files.

Installation

The Log4Ant installation is similar to that of any optional Ant package. The following steps describe how to install and verify Log4Ant in your Ant runtime environment. See the JWare Software website (jwaresoftware.org/wiki/log4ant/) for the latest installation and release notes.

1. If you haven't already done so, download and install an Ant distribution—at least version 1.7.0 or 1.7.1. **Log4Ant does not work with Ant 1.8 nightly builds.** Verify that Ant is properly installed by trying a simple Ant script.
2. Download, verify, and extract a Log4Ant distribution. We suggest you download the “_withdeps” binary distribution that includes all the required third-party libraries Log4Ant needs—you can use the sample “log4ant-install-check.xml” script to verify the installation is working properly.
 - F If you must generate all binaries for your environment, download the source-only distribution. The included “ez-build.xml” Ant build file can generate a default distribution from the source. Be sure you configure the “ez-build.properties” file for your environment. You must use JDK 1.5 or newer to build the Log4Ant package.

In the remaining steps we will use `<LOG4ANT_DIR>` to refer to the directory into which the Log4Ant distribution was extracted or built.

3. Download, verify, and install a logging system implementation. SLF4J is a simple façade to a variety of existing logging systems. You will need to select one of these to use. Some of the popular logging systems that SLF4J works with out-of-the-box include Apache Log4J and Qos.ch's LOGBack.
4. Update your Ant runtime environment to include the Log4Ant jar file `<LOG4ANT_DIR>/lib/jw-log4ant.jar` and all of its dependencies in your Ant classpath. There are several ways of telling Ant about third-party jar files; the easiest method is to copy the jar files into your Ant distribution's `lib` directory. A safer approach is to install Log4Ant in its own location and update the `CLASSPATH` used when you run Ant (for example, by using the `-lib` option to run Ant or a custom `$HOME/.antrc` file).
5. Verify the Log4Ant tasks are accessible from Ant. The easiest way to do this is to run Ant against the starter “log4ant-install-check.xml” script from the distribution. From the `<LOG4ANT_DIR>/etc` directory, run ‘`ant -f log4ant-install-check.xml`’. This sample script loads the Log4Ant antlib. If Ant is unable to locate the Log4Ant classes, even this simple script will fail.
6. Read the rest of this User Guide for an overview of the Log4Ant components.
7. Using the install check script as a guide, include the Log4Ant antlib in your script file(s) using a standard `<taskdef>`.
8. Start using Log4Ant!

Log4Ant Components

The remaining sections describe, in detail, the main components in the Log4Ant antlib. The recommended namespace URI and prefix for Log4Ant is “jware.log4ant” and “emit:” respectively; the component documentation assumes these values.

The JWare Software website contains additional reference documentation; in particular, read the Log4Ant Properties page for the list of OS and Java properties that control Log4Ant. And if you're a developer looking to extend or use Log4Ant in your own Ant components, you will find additional implementation details online in the Log4Ant Javadoc API reference.

Primary Components

Component	Description
<emit:configuration>	Lets you define shareable Log4Ant configuration like logger hierarchies, logging levels, etc.
<emit:manage>	Lets you install a fall-back or default configuration for all Log4Ant components.
<emit:overlay>	Lets you install a configuration for the duration of a set of enclosed tasks.
<emit:capturelogs>	Lets you capture and redirect logged messages for any Ant-based component. Meant as a bridge to capture output of existing Ant scripts.
<emit:show>	Lets you emit a message and other information directly to your logging system. Replacement for standard <echo> and AntXtra's own <show> component.
<emit:checkpoint>	Lets you emit a shorthand “health beat” message directly to your logging system.
<emit:includes>	Lets you define a shareable set of fixture information that you want captured and sent to your logging system. Supports properties, variables, references, resource collections, etc.
<emit:mappings>	Lets you organize captured messages from Ant-based components before sending to your logging system.
<emit:libcheck>	Gives you the version information for the active Log4Ant antlib including information about the SLF4J implementation being used.
<i>AntToSlf4jConduit</i>	Ant execution listener. Lets you install a Log4Ant based Ant listener for the duration of your entire Ant execution cycle.

LibCheckTask

Class: org.jwaresoftware.log4ant.fixture.LibCheckTask

<emit:libcheck>

Category: Diagnostics

Description

The **LibCheckTask** task (defined `<emit:libcheck>`) is a diagnostics task that lets you determine the active versions of Log4Ant and SLF4J APIs. Calling `<emit:libcheck>` generates three properties: one for Log4Ant (`log4ant.label`), another for SLF4J APIs (`slf4j.label`), and another for the SLF4J implementation in effect (`slf4j-impl.label`). After using the task, your Ant script can display or evaluate the values of these properties.

Parameters

Attribute	Description	Required
prefix	A prefix to prepend to the builtin property names.	No

Nested Elements

The `<emit:libcheck>` does not support any nested elements.

Examples

This snippet displays the current Log4Ant and SLF4J labels which includes version and other product information:

```
<emit:libcheck/>
<echo level="info" message="  Log4Ant:  ${log4ant.label}"/>
<echo level="info" message="SLF4J-apis:  ${slf4j.label}"/>
<echo level="info" message="SLF4J-impl:  ${slf4j-impl.label}"/>
```

See Also

- The `<vendorinfo>` AntXtras task lets you extract very detailed information about the loaded Log4Ant ; for example, you could extract just the version number to use in a condition check.

EmitIncludeItemSet

Class: org.jvaresoftware.log4ant.includes.EmitIncludeItemSet

`<emit:includes>`

Category: Fixture-Control (Data)

Description

The `EmitIncludeItemSet` type (defined `<emit:includes>`) lets you describe a collection of Ant fixture elements that Log4Ant should passthrough to SLF4J as string-based key-value context. When you define an `<emit:includes>` you can include common fixture like project properties and references, or AntXtras variables. You can also include your own name-value pairs, exceptions (as captured by AntXtras protected tasksets), and the contents of any valid Ant resource collection. You use `<emit:includes>` to add *custom* information to each emitted message's standard *MDC* from where your logging system can read and use the information.

By itself, an `<emit:includes>` object is not useful. You will need to refer to it from another Log4Ant component like an `<emit:show>` task or a `<emit:configuration>` data object.

Parameters

Attribute	Description	Required
<code>refid</code>	Reference to another <code><emit:includes></code> declaration to use for definition. If defined, no other parameter is allowed.	No

Nested Element: `<include>`

The `<include>` element describes a fixture element that should be included with the emitted message as part of its *MDC*. Because SLF4J *MDC* only permits string-based information; any referenced fixture object will be transformed into a string representation before it's added to the *MDC*. If a named element does not exist (for example a property that is undefined), Log4Ant omits that item—*nothing* is passed in the *MDC* under that name!

Attribute	Description	Required
<code>property</code>	The name of a single property definition to include.	Yes; one of these.
<code>properties</code>	The comma-delimited names of property definitions to include.	
<code>propertyset</code>	The id of an existing <code><propertyset></code> to include.	
<code>var[iable]</code>	The name of a single variable definition to include.	
<code>variables</code>	The comma-delimited names of variable definitions to include.	
<code>reference</code>	The name of a single reference value to include; the refer-ed to item's value is translated to a string if necessary. This item should exist when the information is read (not when the includes set is defined).	
<code>references</code>	The comma-delimited names of reference values to include.	
<code>name</code>	The name of a custom inline name-value pair to include. Requires a matching 'value' attribute; ignored otherwise.	
<code>value</code>	The value of a custom inline name-value pair to include. Requires a matching 'name' attribute; ignored otherwise.	

Attribute	Description	Required
thrown	The id of a captured script error or AntXtras errors snapshot to include. This item should exist at the time when the information is read (not necessarily when the includes set is defined).	
includesref	A id to another <code><emit:includes></code> to be included in this set.	

Examples

The following example declares a base set of information that should be included by any emitted messages (this requirement would need enforcement via another component).

```
<emit:includes id="always.included">
  <include property="package.label"/>
  <include var=".status"/>
</emit:includes>
```

The following example shows how the inclusion set declared above could be used by an error handling taskset to include baseline fixture details as well as the information relevant to the error. The `<emit:show>` task is one component that can use predefined `<emit:includes>` objects.

```
<oja:iferror capturethrown="last.error" quiet="yes">
  <emit:show messageid="err.generic" level="error">
    <include includesref="always.included"/>
    <include thrown="last.error"/>
  </emit:show>
</oja:iferror>
```

The following example declares a default logging `<emit:configuration>` that has a builtin `<emit:includes>` set. So, as an alternative to declaring a separate data object, you could embed the default inclusions here.

```
<emit:configuration id="default.logconf" isdefault="true" ...>
  <include property="package.label"/>
  <include var=".status"/>
</emit:configuration>
```

See Also

- The `<emit:configuration>` lets you define a full set of log system configuration including a set of default fixture information for each emitted message.
- The `<emit:show>` task has a builtin includes set to add custom context information for just that particular message.

EmitConfigurationType

Class: org.jwaresoftware.log4ant.fixture.EmitConfigurationType

<emit:configuration>

Category: Fixture-Control (Data)

Description

The `EmitConfigurationType` type (defined `<emit:configuration>`) lets you define a shareable collection of log system configuration instructions that other Log4Ant components like `<emit:capturelogs>` and `<emit:show>` can use. An `<emit:configuration>` object is thread-safe after its created so you can refer to a single instance from concurrent threads and sub-projects.

You can define an `<emit:configuration>` so it is applied relative to the thread context of the running execution cycle; this way, you can create a log hierarchy (as represented by composite logger names) in your Ant scripts without knowing the exact logger details of the enclosing context. A macrodef could, for example, setup a logger relative to whatever source called it without knowing exactly what that is.

For a configuration object, any attribute that is not defined, or that is defined as “inherited”, assumes the value returned by either the the execution cycle’s default configuration or the configuration referred-to by the configuration’s own “defaults” parameter.

Parameters

Attribute	Description	Required
refid	Reference to another <code><emit:configuration></code> declaration to use for definition. If defined, no other parameter is allowed.	No
defaults	Reference to another configuration declaration to use for any undefined parameters.	No
to	The immediate logger represented by this configuration. The configuration’s final logger is a combination of this parameter and the “wrt” parameter if defined.	No
wrt	Sets the configuration’s logger (“to”) as relative to something else. The “something else” can be one of the following: “root” or <i>empty</i> “to” is a top-level logger name. “default” “to” is relative to the configuration’s default configuration (shared or specific). “enclosing” “to” is relative to the enclosing execution cycle’s thread context configuration. “\$<scheme>:” “to” is relative to a timestamp defined by a standard datetime function shortcut scheme (obtained when item <i>configured</i>). “enclosing” “to” is relative to the enclosing execution cycle’s thread context configuration. <i>anything else...</i> “to” is relative to the named log configuration reference.	No

Attribute	Description	Required
echo	Set to “yes” to also echo the information (in a compatible format) to the Ant logging infrastructure. Possible values are “yes”, “no”, or “inherit”.	No
[noise]level	The noise level of all emitted messages. One of “fatal”, “error”, “warning”, “info”, “verbose”, or “debug”. If echo is enabled, this level might be translated to a “best-fit” level for the builtin Ant logging system.	No
timestampformat	The format string for all emitted timestamp strings. The format string value can be one of the known AntXtras datetime function shortcuts (e.g. “\$isodatetime:”) or any format accepted by the standard <tstamp> task.	No
isdefault	Set to “yes” to make this configuration the default for the current execution cycle (read <emit:manage> for more information on default configurations).	No; defaults “off”

Nested Element: <include>

The <include> element lets you specify a single fixture item that Log4Ant will include as part of the *MDC* linked with any message that is emitted using this configuration. Because *MDCs* only permit string-based values, Log4Ant will convert the referred-to fixture element to some string representation; for example, a referenced data object’s “toString” method will be used to convert it while a plain Ant property will be used as-is. The <include> item’s format is exactly the same as that used for the Log4Ant’s <emit:includes> data type; read that component’s description for details.

Examples

The following example declares a default emit configuration for an Ant build script. The configuration defines (in human-friendly-speak): all messages are grouped or identified relative to the logger “AntXtras.Builds.Nightly”; if undefined, an emitted message’s noise level will be “info”; all messages should be echoed to the standard Ant logging system also; and, this configuration should be installed as the default fixture configuration.

```
<emit:configuration id="default.logconf"
  to="AntXtras.Builds.Nightly"
  noiselevel="info"
  echo="yes"
  isdefault="yes"
/>
```

The following example declares an emit configuration that uses the one declared in the previous example as its default (any unspecified attributes, other than ‘isdefault’ are inherited).

```
<emit:configuration id="apidocs.logconf"
  defaults="default.logconf"
  to="Documents.Javadocs"
  wrt="default"
/>
```

The following example declares an emit configuration that operates relative to the executing thread's configuration context. Emitted messages are not echoed to the Ant log system.

```
<emit:configuration id="reports.logconf"  
  wrt="enclosing"  
  to="Documents.Reports"  
  echo="no"  
>
```

Note that if an emit-aware task uses this configuration while “default.logconf” (see first example) is the global default configuration, its messages will be sent to the logger: “AntXtras.Builds.Nightly.Documents.Reports”.

See Also

- The `<emit:includes>` lets you define a set of fixture information to include with a message independent of an `<emit:configuration>` object.
- The `<emit:overlay>` taskset lets you apply configuration to a small set of nested tasks.
- The `<emit:manage>` task lets you install a configuration as either the global fallback configuration or the default for the current thread.

EmitManageTask

Class: org.jwaresoftware.log4ant.fixture.EmitManageTask

<emit:manage>

Category: Fixture-Control

Description

The `EmitManageTask` task (defined `<emit:manage>`) lets you install and uninstall global and thread-specific fallback logging configuration. You'll have to define the actual configuration in a separate `<emit:configuration>` object.

In Log4Ant, the term "default configuration" really represents a *LIFO list of configurations* plus one "fall back". The list is comprised of configurations you install with `<emit:manage>` and other configuration tasksets; each newly installed configuration is put at the front of the list and gets to respond first if any Log4Ant component asks for the current "default configuration". If the list is empty (you never installed anything) or no configuration in the list has overridden the requested attribute, then the 'fall back' configuration gets to respond.

Parameters

Attribute	Description	Required
action	Sets the action the manage task should perform; one of "install-fallback", "uninstall-fallback", "install", and "uninstall".	No; defaults "install-fallback"
haltiferror	Set to "no" to prevent task from throwing a script error if a problem occurs (like the named configuration does not exist).	No; defaults "yes"
feedback	Sets the level of feedback for the manage task; one of "none", "veryquiet", "quiet", "verbose", and "normal" (default).	No; defaults "normal"

Nested Element: <parameter>

The generic `<parameter>` element lets you select the logging configuration object the manage task should use. Only the name attribute is interpreted.

Attribute	Description	Required
name	Set to the refid of the <code><emit:configuration></code> object to install or uninstall.	Yes for install actions

Examples

The following example installs a previously defined logging configuration "default.logconf" as the global fallback configuration.

```
<emit:manage action="install-fallback">
  <parameter name="default.logconf" />
</emit:manage>
```

The following example macrodef removes a previously installed thread-specific logging configuration “`compilers.logconf`” in addition to other clean-up functions. If the named configuration was not installed (or removed) the script does nothing.

```
<macrodef name="cleanup-compilers">
  ...
  <emit:manage action="uninstall" haltiferror="no" feedback="none">
    <parameter name="compiler.logconf"/>
  </emit:manage>
  ...
</macrodef>
```

See Also

- The `<emit:overlay>` configuration taskset lets you easily install and uninstall a local configuration for a set of nested tasks.
- The `<emit:configuration>` type lets you define your logging configuration.

EmitOverlayTaskSet

Class: org.jwaresoftware.log4ant.fixture.EmitOverlayTaskSet

<emit:overlay>

Category: Fixture-Control

Description

The `EmitOverlayTaskSet` taskset (defined `<emit:overlay>`) lets you define and apply logging configuration for its nested tasks including those tasks triggered from calls to other targets and/or those triggered from Ant calls to sub-scripts. Because `<emit:overlay>` is an `AntXtras` taskset, it affects messages only for its nested tasks so you can use it to apply different emit configuration to different groups of tasks.

Your Ant script can define the logging configuration using individual taskset parameters, or through a previously declared `<emit:configuration>` reference (the preferred manner). Most of the `<emit:overlay>` parameters have a direct counterpart in the `<emit:configuration>` type.

Nested `<emit:overlay>` tasksets use delegation to search for options— if the configuration nearest to the client task does not define the requested option, the preceding `<emit:overlay>` is queried. This delegation continues until an explicit setting is located or there are no more active `<emit:overlay>`s to ask. Then, the nearest configuration's default configuration is queried. To block this delegating query behavior, set the nearest configuration's `inheritance` parameter to "off".

`<emit:overlay>` is essentially a single-threaded task; bad things will occur if a nested task calls back to the enclosing `<emit:overlay>`'s target from either another thread or from the same thread. Both these cases are checked when a `<emit:overlay>` taskset is verified before it is executed; a script error is thrown if duplicate instances of the same `<emit:overlay>` would end up on the run stack.

Parameters

Attribute	Description	Required
<code>with</code>	Reference to an <code><emit:configuration></code> object to use for logging information. If defined, no other parameters are allowed.	No
<code>to</code>	The immediate logger represented by this overlay's configuration. The final logger is a combination of this parameter and the <code>"wrt"</code> parameter if defined.	No
<code>inheritance</code>	Set to "off" if overlay's configuration should ignore other enclosing overlaid tasksets and use its default information for undefined parameters directly.	No; defaults "on"
<code>[noise]level</code>	The default noise level of all emitted messages. One of "fatal", "error", "warning", "info", "verbose", or "debug". If echo is enabled, this level might be translated to a "best-fit" level for the builtin Ant logging system.	No
<code>echo</code>	Set to "yes" to automatically echo all messages (in a compatible format) to the Ant logging infrastructure. Possible values are "yes", "no", or "inherit".	No

Attribute	Description	Required
wrt	<p>Sets the overlay's logger ("to") as relative to something else. The "something else" can be one of the following:</p> <ul style="list-style-type: none"> "root" or <i>empty</i> "to" is a top-level logger name. "default" "to" is relative to the configuration's default configuration (shared or specific). "enclosing" "to" is relative to the enclosing execution cycle's thread context configuration. "\$<scheme>:" "to" is relative to a timestamp defined by a standard datetime function shortcut scheme (obtained when item <i>configured</i>). "enclosing" "to" is relative to the enclosing execution cycle's thread context configuration. <i>anything else...</i> "to" is relative to the named log configuration reference. 	No
timestampformat	The format string for all emitted timestamp strings. The format string value can be one of the known AntXtras datetime function shortcuts (e.g. "\$isodatettime:") or any format accepted by the standard <tstamp> task.	No

Nested Elements

All available tasks can be nested within an <emit:overlay>'s body including other <emit:overlay> tasksets.

Examples

The following example defines a full set of options for a single <emit:overlay> taskset. All nested emit-aware tasks will use this configuration:

```
<target name="regular-jar">
  <emit:overlay
    to="builds.nightly.libraries"
    level="verbose"
    echo="yes">
    [...your tasks here...]
  </emit:overlay>
</target>
```

The following example first creates a custom logging configuration then uses it as the definition for an target-level <emit:overlay> taskset. Note that the final logging configuration is the same as the previous example for nested tasks.

```
<emit:configuration id="compiler.logconf"
  to="builds.nightly.libraries"
  level="verbose"
  echo="yes"
/>
...
<target name="regular-jar">
  <emit:overlay with="compiler.logconf">
    [...your tasks here...]
  </emit:overlay>
</target>
```

See Also

- The `<emit:capturelogs>` taskset lets you capture regular Ant log messages and redirect them to an external logging system .
- The `<emit:configuration>` type lets you keep your logging configuration separate from the configuration tasksets that might use it.

EmitCaptureLogsTaskSet

Class: org.jvaresoftware.log4ant.capture.EmitCaptureLogsTaskSet

<emit:capturelogs>

Category: Feedback

Description

The EmitCaptureLogsTaskSet taskset (defined <emit:capturelogs>) lets you capture and broadcast *standard* Ant log messages to an external logging system linked by SLF4J. In essence, <emit:capturelogs> lets you treat standard Ant log messages as if they were part of the Log4Ant feedback system—you *no longer have to create a custom Ant listener or Ant logger to connect Ant log messages to an external logging system.*

Because <emit:capturelogs> is an AntXtras taskset, it captures and redirectes messages only for its nested tasks; this implementation lets you use different logging configurations for different groups of tasks.

In order to form a complete bridge to the external logging system, <emit:capturelogs> has to perform two important functions: first, it needs to capture the Ant log message to “cc” it to the external logging system (preserving as much source context as possible), and second, it needs to figure out the best log system grouping or *logger* for the Ant messages.

While <emit:capturelogs> uses the standard Ant listener mechanism to capture logged messages, in order to map these messages to an external logging system in a custom way <emit:capturelogs> needs some help from you in the form of script-defined filters called *logger mappings* (read the next section <emit:mappings>). The <emit:capturelogs> will determine a logged message’s logger from a combination of the message’s source (project, target, task, etc.) , the message’s contents, and, if supplied, your mappings.

To ensure that <emit:capturelogs> works out-of-the-box—without any script-supplied mappings—Log4Ant always imposes an immutable root grouping, called an *indicator*, based on the message’s noise level; it is with respect to this grouping that all other <emit:capturelogs> organization is applied. There are three indicator groupings: “*Problems*” containing noise levels fatal, error, and warning, “*Status*” containing noise level info, and “*Diagnostics*” containing noise levels verbose and debug. Unless you provide custom logger mappings, the <emit:capturelogs> taskset will organize broadcast messages in the following manner: [*indicator*].[*project*].[*target*]. Except for the *indicator*, you can remove any of the individual elements using the `includes` parameter.

You should define your own logger mappings to help <emit:capturelogs> match messages to your preferred message organization scheme, particularly if you use the <emit:show> task to send other custom log messages. (Note that you can change the names assigned to the built-in indicator groupings, but you cannot exclude these elements from the broadcast message’s grouping.)

Parameters

Attribute	Description	Required
with	Reference to an <emit:configuration> to use for broadcasting messages. If not defined, the taskset uses the current default configuration. Note that only the configuration’s logger information is used, other settings like the noise level are ignored.	No

Attribute	Description	Required
mappings	Reference to an <code><emit:mappings></code> to use for matching messages to loggers. If not defined, the default loggers (with <code>includes</code> filters applied) is used.	No
includes	Shorthand filter string for message's grouping. This string can be in one of three forms: the string "all" followed by one or more "- <i>field</i> " where <i>field</i> is one of "project", "target", or "task"; the string "all" followed by one or more "+ <i>field</i> " where <i>field</i> is one of "project", "target", or "task"; a comma-delimited list of <i>fields</i> .	No; defaults "project ,target"

Nested Elements

All available tasks can be nested within an `<emit:capturelogs>`'s body including other `<emit:capturelogs>` tasksets. If `<emit:capturelogs>` is nested, logged Ant messages are copied by *each* active `<emit:capturelogs>`.

Examples

The following snippet copies any messages logged within the `<emit:capturelogs>` and `</emit:capturelogs>` element to a external logging system; the default logger configuration combined with the overlaid "Nightly.Tests" logger will be used:

```
<emit:overlay to="Nightly.Tests">
  <emit:capturelogs>
    [...other Ant task calls here...]
  </emit:capturelogs>
</emit:overlay>
```

The following snippet creates a custom logging configuration then uses that configuration from within a `<emit:capturelogs>` and `</emit:capturelogs>` taskset:

```
<emit:configuration id="tests.logconf"
  to="ProgrammerTests.${$isodate:}"
  echo="off" level="verbose"/>

<emit:mappings id="tests.loggers">
  <mapping type="project" like="*" logger="Log4Ant"/>...
</emit:mappings>

<emit:capturelogs with="tests.logconf" mappings="tests.loggers">
  [...other Ant task calls here...]
</emit:capturelogs>
```

The following snippet organizes all captured log messages in groups named after the message's source project and target (task and other information is omitted):

```
<emit:capturelogs includes="project,target">
  [...other Ant task calls here...]
</emit:capturelogs>
```

See Also

- The `<emit:mappings>` type lets you create custom message to logger mappings in addition to the default `[indicator].[project].[target]` hierarchy.

EmitMappings

Class: org.jwaresoftware.log4ant.capture.EmitMappings

<emit:mappings>

Category: Fixture-Control (Data)

Description

The EmitMappings type (defined <emit:mappings>) lets you define a set of mappings between *standard* Ant log messages and an external logging system's grouping hierarchy typically a *logger*. The various log capturing tasks like <emit:capturelogs> can use these mappings to broadcast captured Ant message events with the appropriate log system configuration. Mappings can be defined to match a single message or collections of messages (using regular expressions or message source categories).

Parameters

Attribute	Description	Required
refid	Reference to another <emit:mappings> declaration. If defined, no other parameter or nested element is allowed.	No
defaults	A reference to another <emit:mappings> that this mapping should use as a fall back if unable to find mapping within itself.	No

Nested Element: <mapping>

The <mapping> element defines a single mapping instruction. Use a message source-based mapping to group entire collections of messages. Use a content-based mapping to select individual messages. If you're mapping task names, be aware that mappings apply only to a task's unqualified name (no namespace prefix is included).

Attribute	Description	Required
type	The message's source category. One of "indicator", "project", "target", "task", or "message".	Yes
name	The value that the message's source must match exactly; for example, if the mapping's source category is "task", this is the name of the task that will match.	Yes; one of these three or the "null" attribute.
names	A comma-delimited list of values that message source can match; the source <i>only has to match one of these names</i> to be considered a match.	
like	The regular expression that the message's source must match; for example, if the mapping's source category is "target", this is the expression that the target's name must match.	
logger	The log system's grouping identifier for all message events matching this mapping.	Yes; one of these two.
loggermatch	The log system's grouping identifier as a result regular expression. Sub-expressions from the like parameter's match are used to replace elements in this expression.	

Attribute	Description	Required
ignorecase	Set to “yes” if the name should be lower-cased before it is compared. Applies to all types of matches.	No; defaults “no”
null	Set to “yes” if this item should <i>match the null value only</i> .	No

Examples

The following example defines a set of logger mappings that relocates the built-in “problems” indicator grouping under a datestamp and further puts all messages under an “Log4Ant” group:

```
<emit:mappings id="base.loggers">
  <mapping type="indicator" name="problems" logger="${DSTAMP}.Problems"/>
  <mapping type="project" like="(*)" loggermatch="Log4Ant\.\1"/>
</emit:mappings>
```

The following example defines a single mapping that matches either the <property>, <copyproperty>, <pathproperty>, or <assign> tasks to the “Fixture.Setup” sub-logger. The mappings inherit default project and indicator mappings from the “base.loggers” mappings defined in the previous example:

```
<emit:mappings id="main.loggers" defaults="base.loggers">
  <mapping type="task" names="property,pathproperty,copyproperty,assign"
    logger="Fixture.Setup"/>
  ...
</emit:mappings>
```

The following example extends the previous one to include a message based mapping. The mapping will group any message that starts with the string “Class ” and contains the marker string “loaded from ” into a sub-logger “Fixture.Setup.ClassLoad”. This filter’s groupings are applied relative to the appropriate *indicator.project.target* grouping:

```
<emit:mappings id="main.loggers" defaults="base.loggers">
  <mapping type="message" like="^(Class ){1}.* loaded from .*"
    logger="Fixture.Setup.ClassLoad"/>
  ...
</emit:mappings>
```

See Also

- The <emit:capturelogs> task captures Ant logs and uses emit mappings to match message events to groupings for external logging systems.

EmitTask

Class: org.jwaresoftware.log4ant.emit.EmitTask

<emit:show>

Category: Feedback

Description

The EmitTask task (defined <emit:show>) is a replacement for the standard <echo> task that lets you send scoped, complex messages to an external logging system without having to create a custom Ant listener. Because it is based on the AntXtras <show> task, the <emit:show> task also includes all the message customization features of that task like support for localized message resources and dynamic message argument replacement. *The full set of task parameters is described in the next section including inherited parameters.*

Although the <emit:show> task supports its own configuration option, you would normally link a collection of related tasks, including <emit:show> calls, with a single log configuration by using either the <emit:manage> task to install a global default logging configuration or the <emit:overlay> task to install a local configuration.

If you do not supply a message (literal or via a message bundle), this task will *still* send a message event to the target logging system with an empty string as its message.

Parameters

Attribute	Description	Required
messageid	Resource string selector of message to emit. Can be used as the message itself if no matches found in any installed message bundles.	No; only one when used.
message	String literal of message to emit; only used if messageid is undefined or doesn't exist in any message bundle.	
arg0,...,arg9	Values to be substituted into a template message at positions {0} through {9}. Position {0} and {1} are automatically defined to be the task's name and its script location. You can override these values.	No
level	Noise level of the message. One of "error", "warning", "info", "verbose", or "debug".	No; defaults to config
to	The log system grouping with which the message is associated. For Log4J, LOGBack, and J2SE systems this identifies the Logger used to send the message.	No; defaults to config
echo	Set to "yes" to also echo the message to the standard Ant log system in an appropriate format. Set to "no" to only send to external log system.	No; defaults to config
with	Set to a reference of an existing <emit:configuration> object. The show task will use this configuration instead of any defaults.	No
if	Property name; this task will emit its message only if the named property is <i>defined</i> .	No
unless	Property name; this task will emit its message only if the named property is <i>not defined</i> .	No

Nested Element: <include>

The <include> element lets you describe any fixture elements that should be included with the emitted message. The format of the <include> item is exactly the same as the <include> element of the Log4Ant <emit:includes> data type; read the <emit:includes> component's description for further details.

Nested Element: <defaultmessage>

The <defaultmessage> element lets you define a larger message for sending. The message's contents is assumed to be all of the body text of this element. Note that this element's information is only used if neither the 'messageid' nor 'message' parameters are defined.

Examples

The following example emits a message to a listening monitor system and the Ant log system (an extension of what a simple <show> declaration would do):

```
<emit:show messageid="cp.precompile.done" echo="yes"/>
```

The following example emits a startup message that includes the name of the current project as a template argument. Note that the script-supplied arguments start at template argument {2} so we can still use the pre-filled {0} and {1} values from the inherited <show> task.

```
[In messages resource bundle file]
msg.starting=Starting build of {2}.
```

```
[In build script]
<emit:show messageid="msg.starting" arg2="${ant.project.name}"/>
```

The following example emits a message with a specific log system grouping. The other emit options (like level and echo) are inherited from any default logging configuration.

```
<emit:show to="Log4Ant.Compiling" message="Compiling main classes"/>
```

The following example emits error information captured by the <iferror> handler of a surrounding <protect> task.

```
<protect ...>
  <iferror capturethrown="the.err">
    <emit:show messageid="err.generic" thrown="the.err" level="error"/>
  </iferror>
  ...
</protect>
```

The following example extends the previous example by selectively including some fixture information with the emitted message:

```
<protect ...>
  <iferror capturethrown="the.err">
    <emit:show messageid="err.generic" thrown="the.err" level="error">
      <include propertyset="all.test.properties"/>
      <include property="svn.revision"/>
    </emit:show>
  </iferror>
  ...
</protect>
```

See Also

- The `<emit:overlay>` taskset lets you define target-specific logging configuration information so each `<emit:show>` task only includes needs to include instance-specific options.
- The `<emit:checkpoint>` lets you broadcast pre-canned, simple “heartbeat” messages to listening monitors.

CheckpointTask

<emit:checkpoint>

Class: org.jwaresoftware.log4ant.emit.CheckpointTask

Category: Feedback

Description

The CheckpointTask task (defined <emit:checkpoint>) is a replacement for the standard <echo> task that lets you emit a simple checkpoint message to listening monitors or to a file.

The message generated by a checkpoint includes meta information in a fixed format that makes it simple to parse from email subject lines, chat posts, and RSS feeds quickly. Single line messages are not enforced but highly recommended. The noise level associated with checkpoints is “info” by default although you can override this setting if you want to.

The full format for the messages sent out by the <emit:checkpoint> task is:

```
[TICK] <RAGU> <TIMESTAMP> - <YOUR MESSAGE TEXT>
```

Where <YOUR MESSAGE TEXT> is your checkpoint message, <TIMESTAMP> is the timestamp of when the message is sent *from Ant*, and <RAGU> is a special status flag that you can include (omitted by default) to represent the “traffic light” status of some process. The format of the final message emitted depends on the parameters you specify for that command.

Parameters

Attribute	Description	Required
messageid	Resource identifier for message template to broadcast or save. This message becomes <i>part-of</i> the standard checkpoint string.	No; only one when used.
message	String literal of message to broadcast; only used if <code>msgid</code> is undefined. This message becomes <i>part-of</i> the standard checkpoint string.	
arg0,...,arg9	Values to be substituted into a template message at positions {0} through {9}. Position {0} and {1} are automatically defined to be the task’s name and its script-file location.	No
status	Set to a <i>RAGU</i> status setting; one of “R” (red), “A” (amber), “G” (green), or “U” (unknown).	No
tofile	File in which message should be saved; file will be created if necessary. If defined, the message is <i>only</i> sent to the file; it is not broadcast to the external log system.	No
append	Set to “yes” if the new message should be appended to an existing file’s contents. (Ignored if <code>tofile</code> option not defined.)	No
with	Reference to an <emit:configuration> to use for broadcasting messages. If not defined, the task uses the installed default config.	No
if	Property name; this task will emit its message only if the named property is <i>defined</i> .	No
unless	Property name; this task will emit its message only if the named property is <i>not defined</i> .	No

Examples

The following snippet issues a simple timestamp-only checkpoint. The timestamp format and the log system grouping is determined by the current default logging configuration setup.

```
<emit:checkpoint/>
```

The logged timestamp-only message might look like:

```
20091021T094125-0500
```

The following snippet issues a simple checkpoint on exit from a “libraries” target:

```
<target name="libraries">
  ...
  <emit:checkpoint message="Build of `${v:nextlib}' completed!"/>
</target>
```

Assuming the `nextlib` variable contained ‘main’ the message might look like:

```
[TICK] [20091021T094125-0500] - Build of 'main' completed!
```

The following snippet issues a status checkpoint after running the “FindBugs” tool:

```
<target name="metrics">
  ...
  <emit:checkpoint status="R" message="FindBug count: ${findbugs.warnings}"/>
</target>
```

Assuming the `findbugs.warnings` property contained 27 the message might look like:

```
[TICK] [R] [20091021T094125-0500] - FindBug count: 27
```

The following snippet issues a timestamp-only checkpoint to a reports file “profiled.last”. The message is not broadcast to any log system:

```
<emit:checkpoint tofile="${reports}/etc/profiled.last"/>
```

See Also

- The `<emit:show>` task can be used for general feedback messages where you decide the format and noise level of the broadcast information.

Log4Ant Antlib Definitions

The table below describes the definitions in the default antlib included with the JWare/AntXtras Log4Ant distribution. To use these definitions, just load the Log4Ant `antlib-ns.xml` file into your Ant runtime. If the Log4Ant jars and all of the required dependency jars are in your Ant runtime's classpath, you can load main components as shown below. We assume you will associate the local "emit:" prefix to the "jware.log4ant" namespace:

```
<taskdef uri="jware.log4ant"
  resource="org/jwaresoftware/log4ant/antlib-ns.xml"/>
```

Component Name	Class Name (<oj> is short for 'org.jwaresoftware')
emit:libcheck	<oj>.log4ant.fixture.LibCheckTask
emit:includes	<oj>.log4ant.includes.EmitIncludeItemSet
emit:configuration	<oj>.log4ant.fixture.EmitConfigurationType
emit:mappings	<oj>.log4ant.capture.EmitMappings
emit:manage	<oj>.log4ant.fixture.EmitManageTask
emit:overlay	<oj>.log4ant.fixture.EmitOverlayTaskSet
emit:capturelogs	<oj>.log4ant.capture.EmitCaptureLogsTaskSet
emit:show	<oj>.log4ant.emit.EmitTask
emit:checkpoint	<oj>.log4ant.emit.CheckpointTask

GNU Free Documentation License

Version 1.2, November 2002

Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. 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.

1. 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 language. (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.

2. 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.

3. 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 redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. 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:

- 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" section. 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.

5. 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 Invariant 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."

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.